
UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

DEPARTAMENTO DE COMPUTAÇÃO

# Explorando Funções na Representação de Materiais em Renderizações Realistas

Trabalho de Conclusão de Curso

Mislene da Silva Nunes


São Cristóvão – Sergipe

2017

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

DEPARTAMENTO DE COMPUTAÇÃO

Mislene da Silva Nunes

# Explorando Funções na Representação de Materiais em Renderizações Realistas

Trabalho de Conclusão de Curso submetido ao Departamento de Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientadora: Beatriz Trinchão Andrade de Carvalho

São Cristóvão – Sergipe

2017

**Mislene da Silva Nunes**

# Explorando Funções na Representação de Materiais em Renderizações Realistas

Trabalho de Conclusão de Curso submetido ao Departamento de Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Trabalho aprovado. São Cristóvão – Sergipe, 17 de Novembro de 2017:

---

**Beatriz Trinchão Andrade de Carvalho**
Orientadora

---

**Leonardo Nogueira Matos**
Convidado 1

---

**Gastão Florêncio Miranda Junior**
Convidado 2

São Cristóvão – Sergipe
2017

*Eu dedico este trabalho à minha família.*

# Agradecimentos

Primeiramente, agradeço a Deus por tudo que tem feito e faz por mim.

Agradeço principalmente a minha mãe, Maria Aparecida, por ser sempre a minha fonte de inspiração pra ir em busca dos meus sonhos, com certeza me tornei a pessoa que sou hoje graças a ela, te amo muito mãe.

Agradeço a meus irmãos, Leonardo, Bruno e principalmente a minha irmã gêmea Mislaine, que sempre esteve me acompanhando nessa jornada.

Agradeço a meus amigos por sempre estarem comigo.

A vida é cheia de surpresas, então Deus me permitiu entrar em uma turma, onde conheci este professor incrível, Gastão Florêncio, que admiro muito como professor e pessoa. Muito obrigada pelo apoio, conselhos, ensinamentos, com certeza a minha formação enriqueceu bastante depois de conhecê-lo.

Agradeço a uma pessoa muito especial pra mim, Beatriz Trinchão, minha orientadora. Desde o meu primeiro período eu já a admirava, e desde então a admiro mais ainda, pois ela é incrível como pessoa, professora e orientadora, é um prazer enorme ser sua aluna e orientanda. Com certeza essa admiração contribuiu muito para que eu alcançasse meus objetivos, muito obrigada! por acreditar em mim, por me incentivar nos estudos, por sempre me ajudar, pela paciência comigo, por tudo! Pode ter certeza que tem uma grande participação sua na minha formação, pois sempre foi uma fonte de inspiração pra mim.

*"If you want something, go get it period."*
*(Will Smith)*

# Resumo

A renderização realista é uma área da computação gráfica que visa simular uma imagem do mundo real. Para que isso seja possível é necessário representar a aparência de materiais com fidelidade ou com a maior aproximação possível. Neste contexto, surgem as funções de distribuição de refletância bidirecional (do inglês, Bidirectional Reflectance Distribution Functions - BRDFs), que são funções que, ao receberem uma iluminação incidente e uma perspectiva, reproduzem a cor que o material apresenta sob esta perspectiva. Desta forma, este trabalho busca explorar funções de refletância na representação de materiais em renderizações realistas, com o objetivo de aumentar o número de funções de refletância disponíveis em uma ferramenta de renderização e de possibilitar um aumento na representatividade de materiais do mundo real. Para alcançar tais objetivos, foi feito um estudo sobre ferramentas de renderização, para a escolha de uma, a implementação de funções não contidas na mesma, assim como também a implementação de novos materiais que utilizam essas funções para caracterizar a forma como eles refletem a luz. Para avaliar os resultados obtidos, um conjunto de cenas foram desenvolvidas, dentre elas uma cena realista, onde comparações entre os materiais inseridos e os já existentes foram realizadas. Além disso, para cenas simples foi mostrado que utilizando os materiais inseridos em geral consumiram menos tempo de renderização e produziram resultados semelhantes aos de cenas construídas com materiais já existentes na ferramenta, como também resultados realísticos.

**Palavras-chave**: Renderização Realista, Funções de Refletância, Materiais, Ferramentas de Renderização.

# Abstract

Realistic rendering is an area of computer graphics that aims to simulate a real-world image. In order to do that, it is necessary to represent the appearance of materials faithfully or as closely as possible. In this context, the Bidirectional Reflectance Distribution Functions (BRDFs), which are functions that receive incident lighting and a perspective, reproduce the color that the material presents under a given perspective. This way, this work seeks to explore reflectance functions in the representation of materials in realistic renderings, in order to increase the number of reflectance functions available in a rendering tool, and to allow an increase in the representativeness of materials of the real world. To achieve this goal, a study was made on rendering tools, to choose one, the implementation of functions not contained in it, as well as the implementation of new materials that use these functions to characterize how they reflect the light. To evaluate the obtained results, a set of scenes were developed, among them a realistic scene, where comparisons between inserted and existing materials were performed. In addition, for simple scenes it was shown that using the inserted materials generally consumed less rendering time and produced similar results to scenes constructed with already existing materials in the tool, as well as realistic results.

**Keywords**: Realistic Rendering, Reflectance Functions, Materials, Rendering Tools.

# Lista de ilustrações

# Lista de tabelas

# Lista de códigos

# Sumário

# 1

# Introdução

Renderização, uma grande área da computação gráfica, é um termo herdado da arte e lida com a criação de imagens com efeitos de iluminação e sombra a partir de modelos computacionais 3D (SHIRLEY; MARSCHNER, 2009). Um dos principais desafios desta área é a criação de imagens que sejam cada vez mais semelhantes com a aparência do mundo real, tornando-se indistinguíveis de fotos tiradas do mesmo. A renderização se tornou indispensável e está presente em diversas atividades, que abrangem desde a produção de filmes ao desenvolvimento de jogos. Além disso, ela proporciona novas direções para a expressão criativa, entretenimento e visualização (PHARR; HUMPHREYS, 2010).

A renderização realista visa reproduzir imagens que sejam idênticas a fotos tiradas do mundo real. Sua principal aplicação é a simulação da aparência de objetos reais e de cenas naturais. Neste contexto, trabalhos publicados nessa área apresentam desde simulações de cenas naturais (DOBASHI; YAMAMOTO; NISHITA, 2001) e renderizações de objetos utilizados no mundo real (LAUR et al., 2006) até simulações das características da fisionomia humana (BORSHUKOV; LEWIS, 2003). Uma aplicação desses trabalhos pode ser vista em filmes de animação que buscam cada vez mais simular a realidade.

Renderizações podem ser feitas por ferramentas que também possibilitam modelagem, animação, efeitos especiais, simulação, criação de jogos, entre outras características. Entretanto, todas essas características nem sempre são encontradas em uma única ferramenta, fazendo-se necessário escolher a mais adequada para a finalidade desejada. Detalhes minuciosos podem ser trabalhados nessas ferramentas, tais como a escolha dos materiais, das texturas, e dos objetos 3D com a possibilidade da deformação dos mesmos.

Um fator importante para a reprodução realista de modelos tridimensionais é a modelagem da aparência, que compreende a representação da forma como os materiais de uma cena refletem a luz em uma certa direção sob diferentes configurações de iluminação. Essa modela-

gem é feita através do uso de modelos de refletância, que são funções que, ao receberem uma iluminação que incide em uma superfície e uma perspectiva, reproduzem a cor que o material apresenta sob esta perspectiva.

Na construção de uma cena realista utilizando uma ferramenta de renderização é essencial o cuidado com a modelagem da aparência, possível através de uma análise do comportamento dos materiais escolhidos para interagir com a iluminação. O foco de uma cena realista é atender a todos ou quase todos os detalhes que são percebidos em uma cena do mundo real. Essas ferramentas proporcionam a flexibilidade de utilizar diferentes materiais e diferentes texturas, que contribuem para a aproximação da aparência de diversos materiais do mundo real.

Para a reprodução do que é visto no mundo real, é necessário entender o papel dos materiais, visto que a aparência de um material essencialmente é uma função da sua interação com a luz. Desta forma, é necessário analisar o comportamento de diferentes materiais ao interagir com a luz. Eles podem refletir a luz, absorver a luz ou podem exibir fenômenos mais complexos, como a reflexão sob a superfície e fluorescência (WEYRICH et al., 2009). Devido a esses diferentes comportamentos, para reproduzir a aparência de diferentes materiais são utilizados diferentes modelos de refletância, que diferem entre si em dimensionalidade e complexidade, dependendo das propriedades do material a ser reproduzido.

Em renderizações realistas a aparência dos materiais é essencial para reproduzir imagens que representem o mundo real. Neste contexto, este trabalho visa explorar funções de refletância na representação de materiais e a inserção de pelo menos uma nova função que não esteja contida em uma ferramenta de renderização, com o objetivo de representar outros materiais do mundo real.

## 1.1 Objetivos

O objetivo deste trabalho é explorar o uso de novas funções de refletância em uma ferramenta de renderização, buscando representar outros materiais do mundo real. Este trabalho é constituído dos seguintes objetivos específicos:

- Encontrar uma boa ferramenta de renderização para ampliar o número de funções de refletância disponíveis pela mesma.
- Possibilitar a representação de uma gama mais vasta de materiais do mundo real.

## 1.2 Metodologia

Para alcançar os objetivos delineados neste trabalho, as seguintes fases foram definidas:

- Revisão sistemática sobre funções de refletância e renderização;

- Pesquisa de ferramentas de renderização;
- Seleção de uma ferramenta de renderização que seja bem estruturada em sua implementação, e que também proporcione um bom resultado visual na renderização;
- Identificação de uma função de refletância que não tenha sido implementada na ferramenta escolhida;
- Implementação dessa função;
- Desenvolvimento de cenas 3D para gerar renderizações realistas;
- Desenvolvimento de experimentos para comparar e verificar a aparência dos resultados obtidos com as renderizações.

## 1.3 Estrutura do Documento

Os capítulos deste documento estão organizados da seguinte forma:

- Capítulo 1 - Introdução: uma visão geral sobre a área e descrição do objetivo do trabalho;
- Capítulo 2 - Conceitos: conceitos essenciais para a compreensão do trabalho;
- Capítulo 3 - Trabalhos Relacionados: trabalhos que serviram como referência para a exploração do tema;
- Capítulo 4 - Aprimorando uma Ferramenta de Renderização: apresentação de experimentos feitos nas ferramentas de renderização, das funções de refletância e materiais implementados na ferramenta escolhida;
- Capítulo 5 - Experimentos e Resultados: apresentação dos experimentos e seus resultados;
- Capítulo 6 - Conclusão: considerações gerais sobre o trabalho.

# 2
# Conceitos

Neste capítulo serão apresentados os conceitos necessários para a compreensão e desenvolvimento do trabalho. A Seção 2.1 mostra conceitos essenciais sobre funções de refletância que descrevem a forma como a luz se dispersa nas superfícies. Em seguida, a Seção 2.2 discorre sobre renderização e os principais conceitos necessários para a construção de uma cena. Por fim, na Seção 2.3 são apresentados conceitos detalhados sobre integradores.

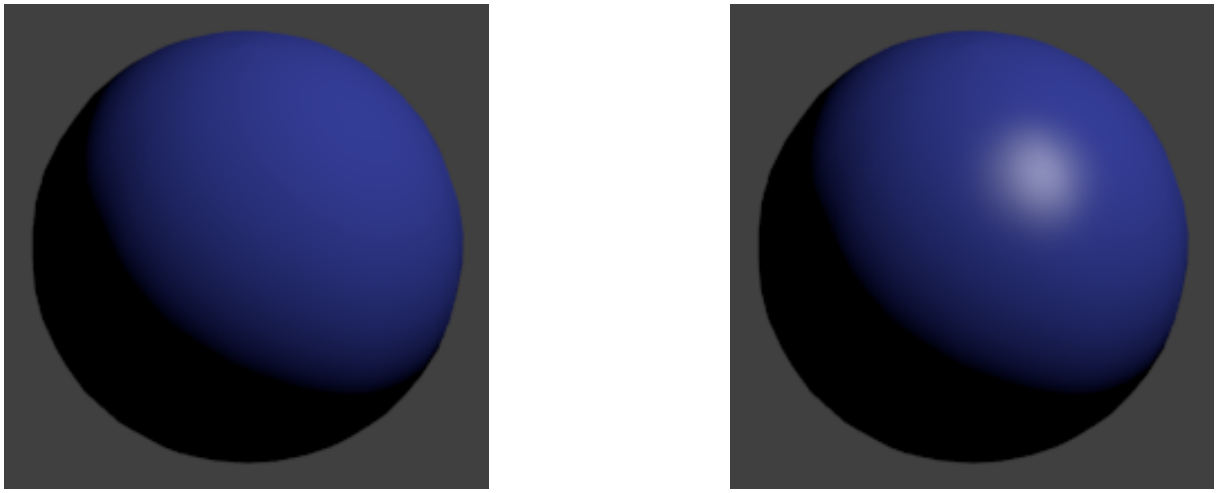
## 2.1 Funções de Refletância

Quando um material interage com a luz, a sua superfície a dispersa, podendo absorver e refletir parte dela. Essa reflexão pode ser modelada pela distribuição espectral da luz refletida e pela sua distribuição direcional. Por exemplo, seja um objeto composto por um material que absorve a maior parte da luz nos comprimentos de onda vermelho e verde; em contrapartida, reflete a maior parte da luz no comprimento de onda azul. Ao iluminar esse objeto com uma luz branca, a cor visualizada é azul, apresentando possivelmente destaques brilhantes de cor mais clara do que azul. A cor exibida por um material é a reflexão da luz que não foi absorvida (PHARR; HUMPHREYS, 2010).

Para materiais com aparência mais reflexiva (como espelho ou superfície polidas), a posição do observador tem bastante importância para a reflexão a partir de um ponto da sua superfície. Analisando um espelho, um ponto em sua superfície fornece diferentes perspectivas de objetos refletidos nesse espelho, à medida que a perspectiva do observador muda (PHARR; HUMPHREYS, 2010).

A Figura 1 mostra os resultados de duas renderizações de um mesmo objeto composto por um material, diferenciando apenas em ser um material puramente difuso ou uma combinação de difuso e especular. Na figura, nota-se que apenas a renderização 1(b) apresenta um destaque

brilhante de cor mais clara do que azul, isso é devido ao uso de um material difuso e especular, que contém uma função de refletância responsável pela reflexão difusa e outra para a reflexão especular. Já a renderização 1(a) não apresenta este comportamento, devido ao uso de um material puramente difuso, que contém apenas uma função de refletância responsável pela reflexão difusa.



(a) (b)

Figura 1 – Renderizações de uma esfera. Renderizada com (a) material difuso e (b) material difuso e especular. Foi utilizada a ferramenta Blender (FOUNDATION, 2017).

### 2.1.1 Funções de Distribuição de Refletância Bidirecional

As funções de distribuição de refletância bidirecional (do inglês, *Bidirectional Reflectance Distribution Functions* - BRDFs) descrevem a refletância de um ponto $p$ em uma superfície através do quociente entre a radiância refletida $L_o$ e a irradiância $E_i$ incidente neste ponto. A irradiância representa a quantidade de luz que incide sobre uma superfície. Já a radiância representa a quantidade de luz que é emitida a partir de uma superfície, onde a luz emitida pode variar de acordo com a direção. Além disso, ela é emitida por unidade de área da superfície, levando em conta um observador ou sensor (WEYRICH et al., 2009).

Em geral, as BRDFs são definidas através das direções de radiância e irradiância, representadas respectivamente por $\omega_o$ e $\omega_i$. Quando as funções são constantes sobre os pontos da superfície, essas direções fazem com que as funções sejam compostas por quatro variáveis:

$$f_r(\omega_i, \omega_o) = f_r(\theta_i, \phi_i, \theta_o, \phi_o) = \frac{\mathrm{d}L_o(\omega_o)}{\mathrm{d}E(\omega_i)}. \tag{2.1}$$

Onde os parâmetros $\theta_i, \phi_i$ e $\theta_o, \phi_o$ são, respectivamente, as coordenadas polares de $\omega_i$ e $\omega_o$. A Figura 2 ilustra os parâmetros de uma BRDF. Para representar mudanças da BRDF sobre a superfície é necessário representar os pontos da mesma. Assim, na Equação 2.1 é inserida a

variável $p$ para representar qualquer ponto pertencente à superfície (CARVALHO, 2013). Neste contexto, a Equação 2.1 passa a ser definida como:

$$f_r(\mathrm{p}, \omega_i, \omega_o) = f_r(\mathrm{p}, \theta_i, \phi_i, \theta_o, \phi_o) = \frac{\mathrm{d}L_o(\mathrm{p}, \omega_o)}{\mathrm{d}E(\mathrm{p}, \omega_i)}. \tag{2.2}$$

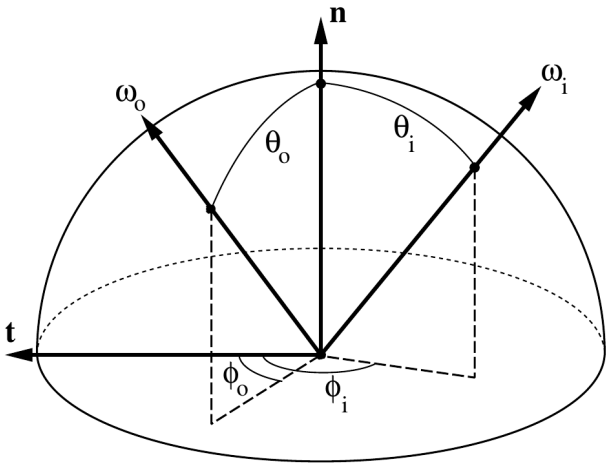



Figura 2 – Parâmetros de uma BRDF: $\omega_i = (\theta_i, \phi_i)$ e $\omega_o = (\theta_o, \phi_o)$ representam as direções de entrada e saída de luz respectivamente, **n** é o vetor da normal da superfície e **t** é o vetor tangente. (fonte: Weyrich et al. (WEYRICH et al., 2009)).

BRDFs fisicamente plausíveis possuem duas propriedades importantes. A primeira é a conservação de energia, ou seja, toda luz que incide em uma superfície deve ser refletida ou absorvida, e durante a reflexão dessa superfície nenhuma luz deve ser criada. A segunda propriedade, chamada reciprocidade de Helmholtz, implica que as BRDFs devem ser inalteradas quando os ângulos de incidência e de saída são trocados um pelo outro (WEYRICH et al., 2009).

A distribuição de luz transmitida pode ser definida usando funções de distribuição de transmissão bidirecional (do inglês, *Bidirectional Transmittance Distribution Functions* - BTDFs). As BTDFs são definidas por $f_t(\mathrm{p}, \omega_i, \omega_o)$ com $\omega_i$ e $\omega_o$ na esfera unitária em torno de p (CARVALHO, 2013). Já o espalhamento da radiância a partir de uma superfície é representado pelas funções de distribuição de espalhamento bidirecional (do inglês, *Bidirectional Scattering Distribution Functions* - BSDFs), definidas pelo quociente entre a radiância de espalhamento e a irradiância incidente, expressadas por $f(\mathrm{p}, \omega_i, \omega_o)$ (ASMAIL, 1991).

### 2.1.2 Modelos de Refletância

Os modelos de refletância são funções que, ao receberem uma iluminação que incide em uma superfície e uma perspectiva, reproduzem a forma como o material reflete a luz. Na Seção 2.1.2.1 são apresentados alguns modelos de refletância da literatura e na Seção 2.1.2.2 são apresentados materiais e modelos de refletância que podem ser utilizados para reproduzir a aparência destes materiais.

#### 2.1.2.1 Modelos da Literatura

Muitos objetos do mundo real possuem a superfície com uma aparência fosca, tais como papel, borracha, entre outros. Esses objetos não apresentam variação de cor significativa com a mudança da posição do observador, sendo considerados objetos lambertianos (SHIRLEY; MARSCHNER, 2009). Para representar a aparência desses tipos de materiais um modelo utilizado é o lambertiano, que é simplesmente uma constante:

$$f_r = const. = \rho/\pi, \tag{2.3}$$

onde $\rho$ representa o albedo difuso: fração de luz que é refletida pela superfície (WEYRICH et al., 2009).

Algumas superfícies são foscas, mas possuem destaques em cores mais claras (picos de especular). Entre os objetos que contêm esses tipos de materiais estão as tintas brilhantes e tintas envernizadas. Os picos de especular na superfície sofrem alterações à medida que o observador muda de posição. Neste contexto, um vetor que represente a direção do observador precisa ser inserido nas equações. Esses destaques são reflexos da luz, sendo às vezes desfocados, e também remetem à cor dessa luz, uma vez que a contribuição da cor da superfície é reduzida.

Um dos modelos usados para representar a aparência de materiais com pico especular é o Phong (PHONG, 1975). Este modelo proporciona uma aproximação dos materiais do mundo real por simular o pico de especular apresentado por eles, sendo definido pela equação:

$$f_r = k_s(r \cdot v)^n, \tag{2.4}$$

onde $v$ representa a direção do observador e $r$ a reflexão do espelho da direção da luz a partir do plano tangente. Os parâmetros $k_s$ e $n$ são responsáveis por controlar respectivamente a intensidade e a concentração do lóbulo especular, que é um lóbulo centralizado em torno da direção da reflexão do espelho ideal para cada ângulo incidente. A região do lóbulo fornece um pico de especular em destaque por conter mais energia do que o restante do domínio. Esse modelo não é fisicamente plausível, pois não respeita a reciprocidade de Helmholtz e nem a conservação de energia (WEYRICH et al., 2009).

Há outros modelos de refletância bastante conhecidos na literatura. Eles podem aproximar funções analíticas para dados medidos das BRDFs tais como Lafortune BRDF (LAFORTUNE et al., 1997) e Ward BRDF (WARD, 1992), podem ser baseados em microfacetas como o Torrance e Sparrow BRDF (TORRANCE; SPARROW, 1967) e Ashikhmin et al. (ASHIKMIN; PREMOZE; SHIRLEY, 2000), e também podem ser mais complexos, por serem baseados em física, por exemplo, o modelo Oren e Nayar (OREN; NAYAR, 1994). Para obter mais informações sobre esses modelos veja o livro escrito por Weyrich et al. (WEYRICH et al., 2009).

#### 2.1.2.2 Materiais do Mundo Real

Existe uma grande variedade de materiais no mundo real, e uma forma de analisá-los é agrupando-os em categorias. Desse modo, características em comum podem ser encontradas, o que permite selecionar modelos de refletância para representar cada tipo de material. Neste contexto, as seguintes informações descritas nesta seção visam apresentar algumas categorias de materiais e modelos de refletância que podem ser usados para representar a aparência desses materias. As informações desta seção são principalmente baseadas no livro de Shirley e Marschner (SHIRLEY; MARSCHNER, 2009).

Dielétricos são materiais transparentes que refratam a luz que incide neles. Exemplos desses tipos de materiais consistem em água, vidros, entre outros. Quando um raio de luz atravessa de um meio para outro, parte dessa luz é transmitida e pode sofrer um desvio na sua direção original, efeito que define a refração. Outro tipo de material que também refrata a luz é o metal, assim como também reflete, no entanto este é capaz de absorver muito rapidamente a luz incidente.

Metais e dielétricos podem possuir superfícies rugosas. Neste caso, eles contêm superfícies microfacetas, que consistem em uma variação de altura onde a distribuição de facetas é descrita estatisticamente (PHARR; HUMPHREYS, 2010). Essas superfícies se comportam especularmente, proporcionando um efeito que pode ser notado em aços escovados, para a categoria dos metais, e em uma folha de vidro no caso dos dielétricos.

Para os dielétricos, existem distinção entre translúcidos e transparentes, pois o primeiro refere-se a superfícies não polidas, fornecendo um borramento na imagem que é refletida e transmitida. Já o segundo possibilita visualizar a imagem refletida e transmitida sem apresentar borramento.

Outra categoria bastante conhecida é a dos materiais difusos, ou seja, sem pico de especular. A grande maioria das superfícies vistas no mundo real pertencem a essa categoria, tal como o gesso. Para a representação destas superfícies pode ser utilizado o modelo lambertiano. Além disso, para a representação da realidade, materiais reais possuem comportamento especular para ângulos rasos. Dessa forma, também é interessante a representação dessa característica para obter um realismo. Esse comportamento pode ser representado por funções que contém o fator de Fresnel (SCHLICK, 1994) em suas formulações.

Além desses materiais, existem aqueles em que as suas superfícies são compostas por camadas ou são dielétricas com partículas embutidas, proporcionando à superfície uma característica difusa (PHONG, 1975). A luz que incide em uma superfície deste tipo pode ser refletida especularmente ou pode ser transmitida. A luz que é transmitida pode ser absorvida ou dispersa de volta para a superfície dielétrica, podendo ou não ser transmitida novamente.

Para cada tipo de material existem modelos de refletância relacionados para modelar sua aparência. Para os metais e dielétricos são sugeridos modelos de reflexão especular. Para os

materiais foscos e materiais especulares, tais como plásticos e madeiras polidas, são sugeridos modelos que possuem o fator de Fresnel em suas formulações, assim como também é sugerido o modelo lambertiano especular, que é uma mistura dos termos especular e lambertiano. No caso em que as superfícies não sejam lisas, é preciso fazer alguma propagação na componente especular. Modelos de camada rugosa são sugeridos nesta situação, um deles é o modelo apresentado por Ashikhmin e Shirley (ASHIKHMIN; SHIRLEY, 2000).

## 2.2 Renderização

A renderização é uma área importante para a computação gráfica, caracterizada pelo processo de converter uma cena tridimensional em uma imagem. Neste contexto, outras áreas da computação gráfica, tais como modelagem geométrica, texturização, são necessárias para o desenvolvimento da cena tridimensional (PHARR; HUMPHREYS, 2010).

Na renderização realista busca-se criar imagens que sejam indistinguíveis de fotos capturadas do mundo real. No entanto, é importante ressaltar que a percepção também é um fator a ser considerado para que um espectador possa identificar se uma imagem é real ou renderizada (FORES; FERWERDA; GU, 2012; HAVRAN; FILIP; MYSZKOWSKI, 2016). Neste contexto, este trabalho visa gerar imagens a partir de renderizações realistas baseadas em física, ou seja, o realismo é considerado fiel às propriedades da física.

### 2.2.1 Opções de Renderização em uma Cena

Esta seção descreve opções de renderização que podem ser definidas em uma cena. Estas descrições são baseadas no livro de Pharr e Humphreys (PHARR; HUMPHREYS, 2010), e na documentação da ferramenta PBRT, apresentada pelos autores deste livro e disponível online (PHARR; JAKOB; HUMPHREYS, 2016).

#### 2.2.1.1 Câmeras

Uma câmera em uma cena 3D define a projeção e configurações de captura da imagem usada para visualizar a cena. Dois tipos de câmera são oriundos de duas projeções clássicas e bastante utilizadas: ortográfica e perspectiva. A câmera ortográfica é baseada na projeção ortográfica, que leva em consideração uma região retangular da cena e a projeta para a face frontal do cubo que define essa região. Essa projeção faz com que linhas paralelas permaneçam paralelas, e preserva a distância entre objetos (ver Figura 3). A câmera perspectiva é baseada na projeção perspectiva que, assim como a projeção ortográfica, projeta um volume de espaço em um plano de imagem bidimensional. Entretanto, o fator de escala dos objetos é inversamente proporcional à distância. Esta projeção difere da ortográfica, de modo a não preservar distâncias ou ângulos, e as linhas paralelas não permanecem necessariamente paralelas (ver Figura 4) (SHIRLEY; MARSCHNER, 2009).

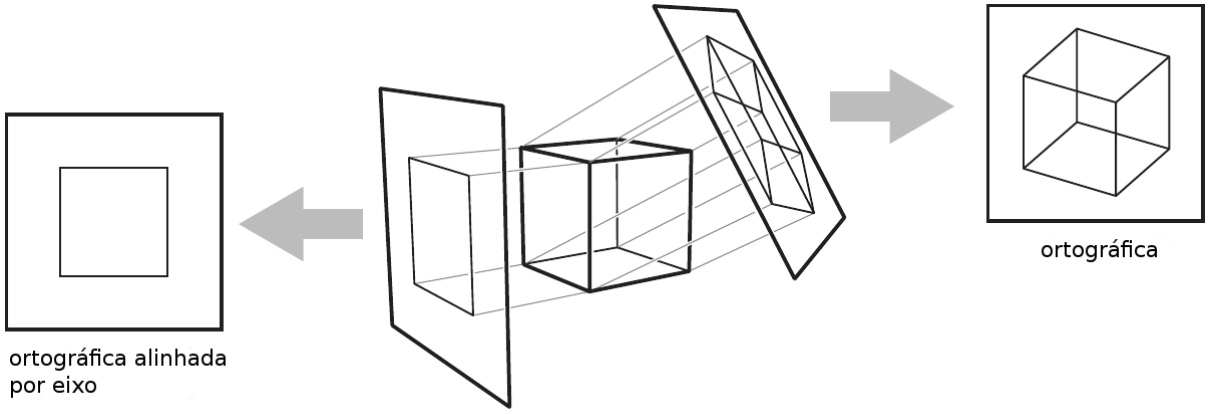



Figura 3 – Aplicação da projeção ortográfica em um objeto 3D. Essa projeção proporciona linhas paralelas e perpendiculares ao plano da imagem. Figura alterada a partir de Shirley e Marschner (SHIRLEY; MARSCHNER, 2009).

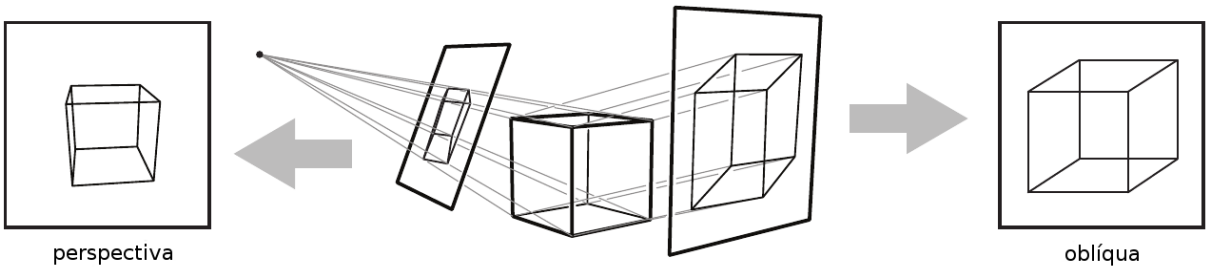
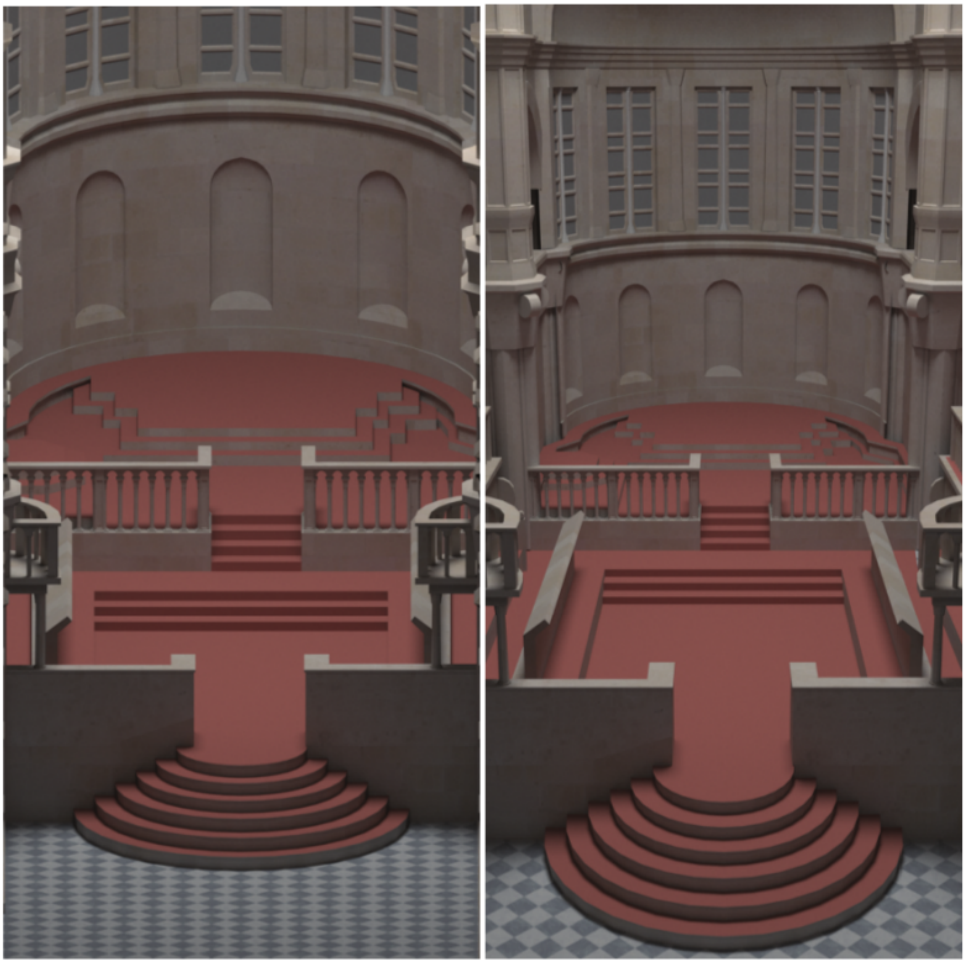



Figura 4 – Aplicação das projeções perspectiva e oblíqua em um objeto 3D. A projeção perspectiva proporciona linhas que passam pelo ponto de vista. Já a oblíqua, é uma projeção paralela que tem o plano da imagem em um ângulo com a direção da projeção. Figura alterada a partir de Shirley e Marschner (SHIRLEY; MARSCHNER, 2009).

Além dessas, existem a câmera ambiente e a câmera realista. A câmera ambiente proporciona uma visão bidimensional de tudo que é visível a partir de um ponto da cena, através do traçado de raios em todas as direções em torno desse ponto. Os algoritmos de traçados de raios seguem o caminho de raios de luz através da cena até que eles cruzem uma superfície. Essa abordagem consiste em um método simples para encontrar o primeiro objeto visível a partir de qualquer posição particular e direção. A câmera realista proporciona imagens geradas a partir de raios de luz que passam por uma simulação de um sistema complexo de lentes.

Exemplos da aplicação das câmeras ortográfica e perspectiva são mostrados na Figura 5. Na figura, é possível perceber a diferença entre esses dois tipos e o efeito que cada uma traz em uma mesma cena. A Figura 6 mostra a aplicação da câmera ambiente que, a partir da posição da câmera, traça raios em todas as direções, gerando uma imagem que representa toda a luz que chega nessa posição na cena.



(a) (b)

Figura 5 – Imagens do Modelo da Igreja. Renderizado com (a) câmera ortográfica e (b) câmera perspectiva. (fonte: Pharr e Humphreys (PHARR; HUMPHREYS, 2010)).

Figura 6 – Modelo San Miguel renderizado com a câmera ambiente (fonte: Pharr e Humphreys (PHARR; HUMPHREYS, 2010)).

#### 2.2.1.2 Amostradores

A teoria de amostragem consiste em tomar valores de amostras discretas de funções em domínios contínuos para usá-las na reconstrução de novas funções que sejam semelhantes à original. Os amostradores tomam como base os princípios da teoria de amostragem, selecionando pontos de amostra no plano da imagem, onde a radiância incidente será calculada. Eles geram amostras tanto para a imagem, quanto para o tempo, a lente e a integração Monte Carlo.

Existem vários geradores de amostras, como por exemplo o amostrador estratificado. Esse amostrador atua dividindo o plano da imagem em regiões retangulares e produz uma única amostra dentro de cada região. A vantagem desse amostrador é que, ao subdividir o domínio de amostragem em regiões que não se sobrepõem, e selecionar uma única amostra de cada região, ele diminui as chances de perda total de características importantes da imagem, pois as amostras são definidas para que não estejam todas próximas. Outros exemplos de amostradores são: adaptativo, melhor candidato, halton, baixa discrepância, aleatório, sobol, entre outros (PHARR; JAKOB; HUMPHREYS, 2016).

#### 2.2.1.3 Filme

Um filme em uma cena 3D especifica as características da imagem que será gerada através do renderizador. O tipo do filme em uma câmera tem bastante impacto sobre a forma como a luz incidente é transformada em cores em uma imagem. O filme estabelece a contribuição da amostra para os *pixels* que estão próximos e atualiza sua representação da imagem, após ter encontrado a radiância para cada raio da câmera. Assim, a imagem final é gravada quando o laço principal de renderização é encerrado.

#### 2.2.1.4 Integradores

Os integradores (ver Seção 2.3 para mais detalhes) calculam a radiância que chega no filme (apresentado na Seção 2.2.1.3) a partir da superfície e mídia participante da cena. Exemplos de mídia participante consistem em fumaça, névoa, poeira, entre outros. A presença de tais fenômenos na cena pode afetar o efeito da propagação da luz ao longo do raio, possibilitando a atenuação da luz através da absorção ou a extinção através do espalhamento em uma direção diferente. Para cenas que simulam um ambiente é importante que o traçador de raios leve em consideração o meio participante para propor mais realismo, uma vez que esses fenômenos interferem na cena. Apesar do PBRT possuir uma implementação que leva em consideração a mídia participante, muitos traçadores de raios a ignoram.

#### 2.2.1.5 Aceleradores

Aceleradores são essenciais para o traçador de raios. Algoritmos para reduzir o número de testes de interseção de raios desnecessários são extremamente úteis. Traçar um único raio

através da cena custa tempo linear no número de primitivas existentes na mesma, pois o raio precisa ser testado para encontrar a interseção mais próxima para cada primitiva. Desse modo, essa atividade acaba se tornando bastante custosa, uma vez que, para a maioria das cenas, o raio não passa perto da grande maioria das primitivas que constituem as mesmas. Neste contexto, os aceleradores são extremamente úteis, pois permitem a rejeição rápida e simultânea de grupos de primitivas e também podem solicitar que as interseções próximas sejam encontradas primeiro e as mais distantes sejam ignoradas.

As interseções raio-objeto podem causar um consumo relevante de tempo de execução em traçadores de raios. Na aceleração de interseção de raios há duas abordagens para solucionar este problema: subdivisão espacial e subdivisão de objeto. Algoritmos de subdivisão espacial atuam decompondo o espaço 3D em regiões e registram quais primitivas estão se sobrepondo em quais regiões. Alguns algoritmos baseiam-se no número de primitivas que se sobrepõem para subdividir as regiões. Para encontrar uma interseção de raios, computam-se as regiões que o raio atravessa e somente as primitivas nas regiões sobrepostas serão testadas para interseção. Já a subdivisão de objetos quebra pouco a pouco os objetos da cena em conjuntos menores de objetos constituintes.

### 2.2.2 Descrevendo a Cena

Esta seção descreve os principais componentes necessários para a construção de uma cena. Após a definição sobre as opções de câmera, renderização e filme, é necessário escolher os materiais, as formas geométricas, luzes e texturas para compor a cena. Além disso, também pode-se utilizar um acelerador para encontrar a interseção de raios de forma mais eficiente. Ao final, o integrador escolhido faz a computação da renderização solicitada para gerar a imagem da cena. Assim como na Seção 2.2.1, as informações descritas nesta seção serão baseadas no livro de Pharr e Humphreys (PHARR; HUMPHREYS, 2010) e na documentação da ferramenta PBRT (PHARR; JAKOB; HUMPHREYS, 2016).

#### 2.2.2.1 Formas

Formas geométricas definem a geometria dos objetos que são colocados na cena. Abstrações para primitivas geométricas são feitas, para que implementem uma interface comum. Essas abstrações proporcionam ao renderizador uma facilidade ao usar essa interface, pois não precisa se preocupar com detalhes das formas. As formas contêm as propriedades geométricas brutas da primitiva, tal como a área da superfície e a caixa delimitadora. Além dessas propriedades geométricas, elas proporcionam uma rotina de interseção de raios.

As formas são definidas no espaço de coordenadas do objeto, como exemplo ao escolher uma esfera, esta é definida em um sistema de coordenadas que tem o seu centro situado na origem. Além disso, nem todas as formas precisam ter a capacidade de determinar se um raio a intercepta. Como exemplo, uma superfície complexa pode ser desenvolvida utilizando triângulos,

que podem ser interceptados diretamente pelos raios. Elas também podem ser usadas como luzes com área, mas para isso é necessário calcular a área da superfície no espaço do objeto.

Algumas formas possuem dois tipos de geometria diferencial em um ponto da superfície: a geometria verdadeira e a geometria de sombreamento. A geometria verdadeira busca refletir com precisão as propriedades da superfície. Por sua vez, a geometria de sombreamento pode ter normais e tangentes diferentes das encontradas na geometria diferencial verdadeira onde, através das normais interpoladas, proporciona malhas de triângulos facetadas com aparência mais suave.

#### 2.2.2.2 Luzes e Luzes de Área

As luzes são responsáveis pela iluminação da cena e são fator determinante na visibilidade dos objetos. Diferentes tipos de luzes podem existir em uma cena, tais como: luzes sem qualquer forma geométrica associada a elas e luzes que descrevem a emissão de uma ou mais formas geométricas, conhecidas por luzes de área.

Diversas luzes podem ser definidas em termos de emissão a partir de um único ponto no espaço, variando a distribuição angular de saída de luz. Um exemplo de luz que possui esse comportamento é a luz pontual, que representa a fonte de luz pontual isotrópica, que emite a mesma quantidade de luz para todas as direções.

Existem outras fontes de luz mais complexas que são baseadas na fonte de luz pontual, tais como holofotes e projeções. Os holofotes são variações da luz pontual. Ao invés de proporcionar iluminação brilhante para todas as direções, eles emitem luz em um cone de direções a partir da sua posição. Há também uma fonte de luz que atua como um projetor de slide, que utiliza um mapa de imagem para projetar sua imagem para fora da cena. A função de projeção em um mapa de imagem possibilita amostrar a distribuição de projeção utilizando técnicas de Monte Carlo. A Figura 7 mostra uma mesma cena renderizada com a luz pontual, holofotes e luz de projeção.

Além dessas, existem as luzes goniométricas e distantes. A luz goniométrica utiliza um diagrama goniométrico que, através de uma fonte de luz pontual, descreve a distribuição angular da iluminação. Essa luz pode representar uma fonte de luz pontal, onde a emissão que é representada por um mapa de textura é direcionalmente variável. A luz distante, também conhecida como luz direcional, atua descrevendo um emissor que define a iluminação na mesma direção para cada ponto no espaço. Essa luz também é conhecida por luz de ponto no infinito, pois à medida que a luz de um ponto se torna cada vez mais distante, ela atua como uma luz direcional.

As luzes de área são fontes de luz definidas com base em formas que emitem luz a partir de sua superfície. Elas são mais complexas do que as luzes apresentadas anteriormente, mas proporcionam sombras mais suaves e efeitos de iluminação mais realistas. Um exemplo de luz de área é a luz difusa, que define uma emissão de radiância uniforme para cada ponto da superfície, levando em consideração todas as direções do hemisfério que estejam em torno da normal da

superfície. É necessário ressaltar que a orientação da normal da superfície é importante.

(a) (b) (c)

Figura 7 – Cena renderizadas sob diferentes fontes de luz: (a) a fonte de luz pontual fornece uma aparência de sombra forte; (b) holofotes cortam a iluminação que está fora de um ângulo definido pelo usuário a partir do eixo central da luz; (c) luz de projeção utilizando um mapa de textura projeta uma imagem em objetos na cena. (fonte: Pharr e Humphreys (PHARR; HUMPHREYS, 2010)).

Um outro exemplo é a luz de área infinita, definida por uma fonte de luz de área infinitamente distante ao redor de toda a cena, emitindo luz em todas as direções. Essa luz é importante para iluminação de ambiente, onde pode-se usar uma imagem que represente a iluminação em um ambiente.

#### 2.2.2.3 Materiais

Materiais são responsáveis pela especificação das propriedades de dispersão da luz das superfícies em uma cena. Como no mundo real existem diversos materiais, padrões dos mesmos são descritos para que possam ser representados por modelos de materiais. A combinação de modelos com texturas (Seção 2.2.2.4) reproduz uma vasta gama de materiais do mundo real.

Para descrever como um determinado modelo de material dispersa a luz é necessário determinar funções de refletância (apresentadas na Seção 2.1) adequadas para a construção do mesmo, pois os materiais podem refletir a luz, absorver a luz, ou apresentar comportamentos mais complexos. Há diversos modelo de material, tais como vidro, plástico, fosco, metal, espelho, cabelo, entre outros.

#### 2.2.2.4 Texturas

Textura pode ser definida como sendo uma função que mapeia pontos em algum domínio tal como um espaço paramétrico de uma superfície (u,v) ou um espaço de objeto (x,y,z), a valores em algum outro domínio tal como espectros ou números reais. Existem vários tipos de funções

que são representadas por textura, como exemplo, funções que retornam uma constante para acomodar superfícies que possuem o mesmo valor de parâmetro em todos as regiões das mesmas. Outro tipo de textura é baseada em mapas de imagem, que são funções bidimensionais que, através de uma matriz de valores de *pixel*, calculam os valores de um ponto específico. Existem também funções de texturas que calculam valores com base em funções.

As texturas são características incorporadas aos modelos dos materiais para a criação de uma grande variedade de aparências dos mesmos. Os materiais possuem diversas características, tais como reflexão difusa, reflexão especular, entre outras. Além disso, os materiais possuem propriedades que variam tipicamente ao longo de uma superfície. Assim, fazer a combinação de texturas com materiais facilita a criação de uma ampla variedade de aparências.

## 2.3 Integradores

Integradores computam a radiância espalhada a partir de superfícies em uma cena. Eles são responsáveis pela avaliação da Equação de Transporte da Luz (do inglês, Light Transport Equation - LTE), representada pela Equação 2.5. Informações sobre interseções entre os raios que saem de uma câmera e a geometria da cena são passadas para integradores de superfície e volume, que computam *shading* e iluminação para calcular a radiância ao longo de um raio. O integrador de superfície representa a luz refletida a partir da primeira superfície visível ao longo do raio. O integrador de volume estima a luz atenuada e dispersa pelo meio participante ao longo do raio, além de afetar a distribuição da radiância. As informações apresentadas nesta seção são baseadas no livro de Pharr e Humphreys (PHARR; HUMPHREYS, 2010).

A LTE descreve a distribuição de equilíbrio da radiância em uma cena. Ela retorna a radiância refletida total em um ponto da superfície em termos da emissão da superfície, de sua BSDF, e da distribuição de iluminação incidente no ponto. Em geral, uma cena contém vários objetos, isso gera uma dificuldade para a LTE, por causa da radiância incidente em um ponto da cena ser afetada pela geometria e propriedades de espalhamentos de todos objetos presentes.

Para impor o equilíbrio energético em uma superfície, a radiância que sai da mesma deve ser igual à radiância emitida com a adição da fração de radiância incidente que está espalhada. A radiância emitida é dada por $L_e$, a BSDF é dada por $f(\mathrm{p}, \omega_i, \omega_o)$, onde p representa o ponto na superfície, e $\omega_i$ e $\omega_o$ são coordenadas polares para direções de incidência e saída. O espalhamento da radiância é dada pela Equação 2.5.

$$L_o(\mathrm{p}, \omega_o) = L_e(\mathrm{p}, \omega_o) + \int_{\mathbb{S}^2} f(\mathrm{p}, \omega_i, \omega_o) L_i(\mathrm{p}, \omega_i) |cos\theta_i| \mathrm{d}\omega_i. \tag{2.5}$$

Um dos integradores mais simples conta apenas com a iluminação direta. A luz direta leva em consideração somente a luz que viaja diretamente da fonte de luz para um ponto sendo sombreado, e ignora a iluminação indireta obtida de objetos presentes na cena que não

são emissivos. Um exemplo de renderização utilizando essa luz é mostrado na Figura 8. Esse integrador possibilita concentrar em alguns dos principais detalhes da iluminação direta, por não ter que se preocupar com a LTE completa. A equação da luz direta retorna a radiância refletida em um ponto da superfície em termos da emissão da superfície, da sua BSDF, e da iluminação direta incidente. A radiância advinda diretamente da fonte de luz é denotada por $L_d(\mathrm{p}, \omega_i)$ e a equação da fonte de luz é definida pela Equação 2.6.

$$L_o(\mathrm{p}, \omega_o) = L_e(\mathrm{p}, \omega_o) + \int_{S^2} f(\mathrm{p}, \omega_i, \omega_o) L_d(\mathrm{p}, \omega_i) |cos\theta_i| \mathrm{d}\omega_i. \tag{2.6}$$

Figura 8 – Modelo renderizado apenas com luz direta. Algumas partes da imagem são completamente pretas porque não são acessíveis diretamente por alguma das fontes de luz. (fonte: Pharr e Humphreys (PHARR; HUMPHREYS, 2010)).

Estratégias são usadas para computar a iluminação direta. Cada método calcula uma estimativa imparcial da radiância de saída em um ponto em uma direção. Uma estratégia é iterar sobre todas as luzes, coletando um número de amostras de cada, somando o resultado. Outra estratégia é coletar uma única amostra de uma das fontes de luz que foi escolhida aleatoriamente. A escolha apropriada dessas abordagens depende da cena que está sendo renderizada. Como exemplo, no caso de imagens onde são obtidas muitas amostras por *pixel*, pode ser mais apropriado usar uma única amostra a partir de uma fonte de luz. Caso contrário, amostras de todas as fontes de luz podem ser uma ótima escolha.

Analisando a Equação 2.6, o valor de $L_e$ é calculado no ponto de interseção, e para estimar a integral sobre a esfera é aplicada a Integração de Monte Carlo, que é um método que utiliza amostragem aleatória para estimar valores das integrais. Olhando apenas uma parte da equação direta:

$$\int_{S^2} f(\mathrm{p}, \omega_i, \omega_o) L_d(\mathrm{p}, \omega_i) |cos\theta_i| \mathrm{d}\omega_i. \tag{2.7}$$

Ao separá-la em um somatório de acordo com as luzes da cena obtém-se:

$$\sum_{j=1}^{luzes} \int_{S^2} f(\mathrm{p}, \omega_i, \omega_o) L_{d(j)}(\mathrm{p}, \omega_i) |\cos\theta_i| \mathrm{d}\omega i. \tag{2.8}$$

Onde $L_{d(j)}$ indica a radiância incidente da j-ésima luz e

$$L_d(\mathrm{p}, \omega_i) = \sum_j L_{d(j)}(\mathrm{p}, \omega_i). \tag{2.9}$$

Estimar cada termo do somatório individualmente, somando os resultados, proporciona uma abordagem válida. No entanto, uma alternativa é estimar a luz direta de apenas uma fonte de luz aleatória, e compensar multiplicando pelo número total de luzes. Esta abordagem é bastante útil em cenas com grande quantidade de luzes, e além disso, a técnica de integração Monte Carlo fornece um cálculo correto em média. A escolha da luz individual pode ser aleatória e por probabilidade. Quanto melhor for a escolha, mais eficiente será o estimador Monte Carlo e menos raios serão necessários para reduzir a variância.

Depois de ter escolhido a luz em particular que será utilizada para estimar a iluminação direta, é necessário calcular o valor da integral:

$$\int_{S^2} f(\mathrm{p}, \omega_i, \omega_o) L_d(\mathrm{p}, \omega_i) |cos\theta_i| \mathrm{d}\omega_i. \tag{2.10}$$

Para essa luz necessita-se escolher uma ou mais direções $\omega_j$ e aplicar o estimador Monte Carlo:

$$\frac{1}{N} \sum_{j=1}^{N} \frac{f(\mathrm{p}, \omega_j, \omega_o) L_d(\mathrm{p}, \omega_j) |\cos\theta_j|}{\mathrm{p}(\omega_j)}. \tag{2.11}$$

Com o objetivo de reduzir a variância, é usada amostragem de importância (do inglês, importance sampling) para escolher as direções $\omega_j$. Ressalta-se que a complexidade existente na BSDF e nos termos de radiância diretas pode causar dificuldades para encontrar distribuições de amostragem que correspondam bem ao seu valor.

# 3

# Trabalhos Relacionados

Esta seção descreve trabalhos relacionados ao desenvolvimento de renderizações realistas (Seção 3.1), ferramentas de renderização que foram estudadas (Seção 3.2) e trabalhos que buscam explorar funções de refletância em ferramentas de renderização (Seção 3.3).

## 3.1 Renderização Realista

A renderização realista busca reproduzir imagens semelhantes a cenas do mundo real. Neste contexto, estão surgindo cada vez mais trabalhos que buscam alcançar tal objetivo. Esses trabalhos apresentam novas abordagens para proporcionar imagens cada vez mais próximas da realidade. Além disso, eles também apresentam aplicações, que vão desde simulações de cenas naturais até a produção de filmes.

Um método para criar imagens realistas de cenas naturais foi apresentado por Dobashi et al. (DOBASHI; YAMAMOTO; NISHITA, 2001). Esse método proporciona a criação de imagens que contêm relâmpagos, a fim de simular cenas artificias com tal elemento. Os autores, além de simular relâmpagos, também se preocupam, na renderização, em levar em conta os efeitos de espalhamento. Esses efeitos são obtidos através das nuvens e partículas atmosféricas que são iluminadas pelos relâmpagos.

Por serem complexos e detalhados, é difícil representar objetos naturais com fidelidade. Neste contexto, Weber e Penn (WEBER; PENN, 1995) apresentam um modelo para criar e renderizar árvores, com ênfase na estrutura geométrica geral da árvore. A geometria da superfície é uma propriedade que deve receber atenção especial para que se obtenha um resultado visual plausível. Isso se deve ao fato de que ela proporciona detalhes mais próximos da realidade quando, durante a renderização, é feito o sombreamento.

Os filmes de animação estão buscando cada vez mais simular à realidade. Iben et

al. (IBEN et al., 2013) apresentam um novo método para a simulação de cabelo cacheado que atende às necessidades artísticas, mantendo a forma helicoidal do cacho durante o movimento, e também atende à demanda de desempenho. O uso deste método pode ser visto no filme Valente (do inglês, Brave), onde o método mostrou a sua robustez e estabilidade em vários movimentos animados e em uma variedade de estilos de cabelo.

Nos filmes Procurando Dory (do inglês, Finding Dory) e Piper foram enfrentados alguns desafios, tais como a renderização de uma grande quantidade de água, e a possibilidade da visualização de criaturas através da água e vidro (CHRISTOPHE; RYUSUKE; HECHT, 2016). Já no filme Carros (do inglês, Cars) foi adicionado o traçado de raios ao renderizador RenderMan, o que proporcionou a adição de muitos efeitos tais como reflexões precisas, detalhamento nas sombras, e a oclusão do ambiente que indica as bordas de uma superfície e a proximidade espacial de objetos (LAUR et al., 2006).

Além dos filmes de animação, a renderização realista também está presente em outros gêneros. Como exemplo, o filme Matrix Reloaded utilizou técnicas de renderização fotorrealista para simular faces humanas, onde foram criadas renderizações fotorrealistas para representar a maioria dos atores principais do filme. Dessa forma, foi possível criar quadros totalmente virtuais para algumas das cenas por meio dessas renderizações (BORSHUKOV; LEWIS, 2003).

## 3.2 Ferramentas de Renderização

Para simular a interação entre os componentes de uma cena e gerar imagens que contêm uma cena descrita computacionalmente, ferramentas de renderização são necessárias. Nesta seção, serão apresentadas ferramentas de renderização que foram pesquisadas neste trabalho.

### 3.2.1 RenderMan

O RenderMan é uma tecnologia de renderização da Pixar, com o objetivo de atender não só aos próprios filmes produzidos pela Pixar, mas também aos desafios cada vez maiores encontrados na animação 3D e efeitos visuais. Esta ferramenta possui *shading* baseado em física, que permite oferecer uma iluminação realista com configuração mínima. Além disso, ela dá suporte à iluminação global traçada com raios múltiplos e ao espalhamento em subsuperfície traçada por raios, características que contribuem para a criação de imagens fotorrealistas. Resultados dessas novas caraterísticas podem ser vistos no filme procurando Dory. Essa ferramenta precisa ser comprada para ser usada, mas recentemente a sua versão não-comercial [1] foi disponibilizada, sendo gratuita para todos os propósitos não comerciais, que incluem avaliações, educação, pesquisa e projetos pessoais.

[1] <https://renderman.pixar.com/view/renderman>

### 3.2.2 Blender

O Blender é um ambiente de criação de cenas 3D. Ele suporta todo o pipeline de modelagem, *rigging*, animação, simulação, renderização, composição, acompanhamento de movimento, e até mesmo edição de vídeo e criação de jogos. A ferramenta [2] é livre e possui código fonte aberto. Renderizações utilizando esta ferramenta foram realizadas para conhecê-la e avaliá-la, os resultados são mostrados na Seção 4.1.1.

### 3.2.3 PBRT

O PRBT é uma ferramenta de renderização introduzida juntamente com o livro Physically Based Rendering (do português, Renderização baseada em física) dos autores Pharr e Humphreys (PHARR; HUMPHREYS, 2010), que introduzem os conceitos e a teoria da renderização fotorrealística. Neste livro, além do conteúdo referente à renderização fotorrealística, são apresentadas explicações sobre a implementação da ferramenta assim como o seu código fonte. Uma documentação (PHARR; JAKOB; HUMPHREYS, 2016) também é disponibilizada, contendo cenas já desenvolvidas que podem ser baixadas. Além disso, a ferramenta é livre e possui código fonte aberto. Resultados de renderizações feitas utilizando esta ferramenta são mostrados na Seção 4.1.2.

## 3.3 Funções de Refletância em Ferramentas de Renderização

Devido à necessidade de representar materiais do mundo real, algumas ferramentas de renderização inserem diversas funções de refletância para caratacterizá-los. Por exemplo, Pharr e Humphreys (PHARR; HUMPHREYS, 2010) desenvolveram uma ferramenta de renderização (PBRT), e escreveram a sua documentação e o livro Physically Based Rendering (em português, Renderização Baseada em Física), onde descrevem várias funções de refletância usadas para caracterizar os materiais disponíveis pelo PBRT, tais como os modelos de refletância: Lambertian (WEYRICH et al., 2009), Oren e Nayar (OREN; NAYAR, 1994), Torrance e Sparrow (TORRANCE; SPARROW, 1967), Blinn (BLINN, 1977) e Ashikhmin e Shirley (ASHIKHMIN; SHIRLEY, 2000).

Esses autores em conjunto com Wenzel Jakob produziram uma nova versão da ferramenta, documentação e livro, onde o material usado para representar BRDFs medidos foi substituído por um material que possui uma representação diferente de BRDFs. Outras ferramentas de renderização, tais como o Blender e RenderMan, também contêm funções de refletância. Essas funções proporcionam aproximações mais precisas dos materiais do mundo real, o que contribui para a criação de cenas realistas.

[2] <https://www.blender.org/>

# 4

# Aprimorando uma Ferramenta de Renderização

Esta seção apresenta renderizações feitas com as ferramentas de renderização estudadas (Seção 4.1), as funções de refletância que foram implementadas na ferramenta de renderização escolhida (Seção 4.2), e os materiais que foram implementados utilizando essas funções de refletância (Seção 4.3).

## 4.1 Experimentos nas Ferramentas de Renderizaçao

Para avaliar algumas das funcionalidades disponíveis pelas ferramentas de renderização, um conjunto de renderizações foi criado para comparar as ferramentas estudadas. A Seção 4.1.1 mostra algumas cenas criadas usando o Blender, e a Seção 4.1.2 mostra algumas cenas criadas usando o PBRT.

### 4.1.1 Renderizações com o Blender

Um conjunto de cenas foi criado utilizando o Blender para adquirir conhecimento sobre esta ferramenta. A Figura 9 mostra o desenvolvimento de uma cena complexa, composta de um quarto e um pinguim dentro dele. Já as Figuras 10 e 11 são cenas simples ilustrando os efeitos de alguns modelos de refletância disponibilizados pela ferramenta. A diferença entre elas é que a Figura 11 utiliza uma modelo de refletância mais complexo. Apesar de apresentar uma documentação detalhada e de ser possível encontrar um grande volume de informação sobre a ferramenta, a sua implementação é pouco didática e mal estruturada. Esses fatores levaram à não escolha dela para ser objeto de estudo neste trabalho, uma vez que demandaria bastante tempo para inserir novas funções de refletância.

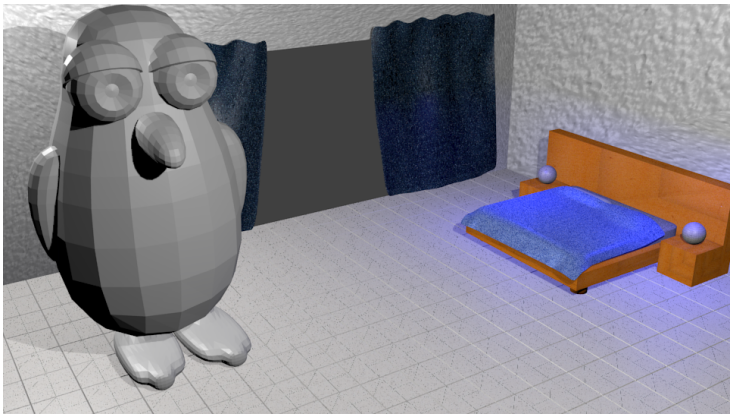
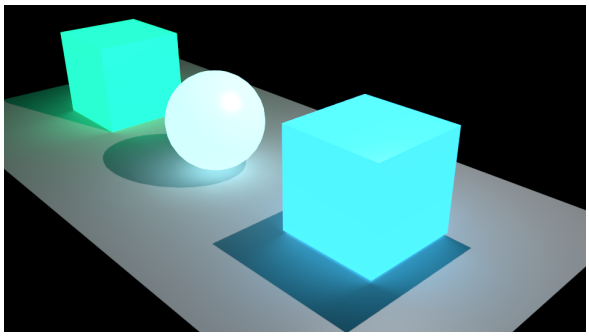
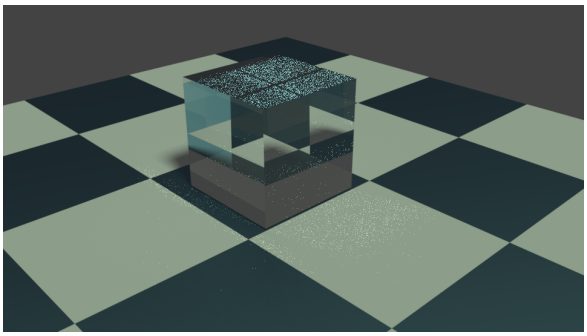


Figura 9 – Modelagem e renderização usando o Blender, baseadas em um tutorial online[1] .



$(a)$



$(b)$

Figura 10 – Renderizações de cenas simples utilizando o Blender. Na Figura (a) foram utilizados os modelos de refletância *Lambertian* e Cook e Torrance. Na Figura (b) é feita uma combinação de diferentes *shaders* e sua renderização utiliza o *cycles render*. Renderizações baseadas em tutoriais online[2,3].

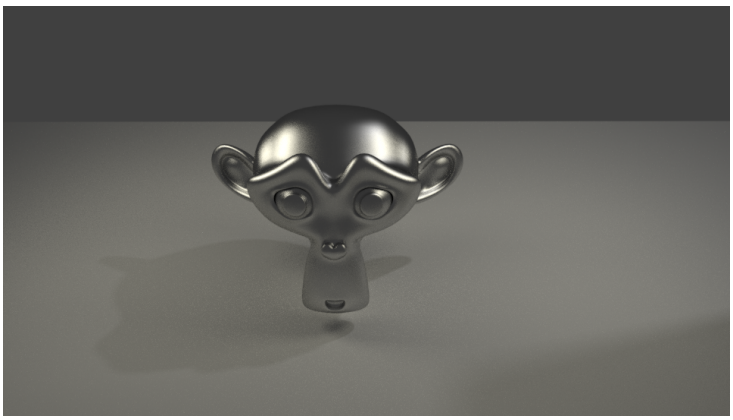


Figura 11 – Renderização utilizando o Blender. Foi usado o modelo de refletância de Ashikhmin e Shirley (ASHIKHMIN; SHIRLEY, 2000) e a funcionalidade *cycles render* da ferramenta. Renderização baseada em um tutorial online[4].

### 4.1.2 Renderizações com o PBRT

Um conjunto de experimentos foi feito utilizando a ferramenta PBRT, para o conhecimento da mesma e analisar a sua viabilidade. Esta ferramenta destaca-se tanto pela simplicidade quanto pela quantidade de informações que são fornecidas, seja por meio do livro de Pharr e Humphreys (PHARR; HUMPHREYS, 2010) ou pela documentação (PHARR; JAKOB; HUMPHREYS, 2016). Desta forma, esta ferramenta foi escolhida para ser objeto de estudo neste trabalho, pois além de disponibilizar informações sobre a mesma, ser livre com código fonte aberto, possui uma implementação bem estruturada na linguagem de programação C++ e de fácil modificação.

A Figura 12 mostra os resultados obtidos a partir da aplicação de diferentes materiais em um mesmo objeto utilizando o PBRT.

(a) (b) (c) (d) (e)

Figura 12 – Renderização de um coelho utilizando o PBRT. Renderizado com o material (a) fosco, (b) *uber*, (c) plástico, (d) metal, e (d) mistura dos materiais fosco e metal.

## 4.2 Funções de Refletância Implementadas

As funções de refletâncias são utilizadas para caracterizar como o material dispersa a luz, podendo este ter diferentes comportamentos ao interagir com a mesma. Neste contexto, quatro funções foram estudadas e implementadas na ferramenta de renderização PBRT-v3 (terceira versão) seguindo a estrutura da mesma:

- modelo de refletância de Cook e Torrance (COOK; TORRANCE, 1982);
- modelo de refletância de Ashikhmin e Shirley (ASHIKHMIN; SHIRLEY, 2000) simplificado por Ngan et al. (NGAN; DURAND; MATUSIK, 2005b);
- modelo de refletância para metais de Brady et al. (BRADY et al., 2014);
- modelo de refletância para dielétricos de Brady et al. (BRADY et al., 2014).

[1] <https://youtu.be/ilQQScEwUGE>
[2] <https://youtu.be/wx4ayy0-VD8>
[3] <https://youtu.be/fgAySB7DoOc>
[4] <https://youtu.be/UTwXG3K4l2g>

A Tabela 1 apresenta os significados dos símbolos em comum nas expressões.

Tabela 1 – Símbolos comuns nas expressões das funções de refletância implementadas. (Fonte: Brady et al. (BRADY et al., 2014)).

| **Notações Comuns** | **Significado** |
|---|---|
| $L$ | vetor da direção da entrada de luz |
| $V$ | vetor da direção do observador |
| $H$ | vetor entre L e V, conhecido por *halfway* |
| $N$ | normal |
| $\delta$ | ângulo entre o halfway e a normal |
| $K_d$ | amplitude do lóbulo difuso |
| $K_s$ | amplitude do lóbulo especular |
| $R_0$ | refletância que incide na normal |
| $\alpha$ | força do pico especular |
| $\beta$ | claridade do pico especular |

Na ferramenta PBRT há muitas funções de refletância do estado da arte implementadas. Neste contexto, foram escolhidas duas funções da literatura não contidas no PBRT para inseri-las. A primeira é oriunda do modelo de refletância apresentado por Cook e Torrance (COOK; TORRANCE, 1982), descrita pela Equação 4.1, que é bastante útil para a simulação de metais e plásticos.

$$
\begin{aligned}
f_r(L,V) &= \frac{K_d}{\pi} + K_s \frac{F}{\pi} \frac{DG}{(N\cdot L)(N\cdot V)} \\
D &= \frac{1}{m^2 \cos^4 \delta} e^{-[\tan(\delta)/m]^2} \\
G &= min\left\{1, \frac{2(N\cdot H)(N\cdot V)}{(V\cdot H)}, \frac{2(N\cdot H)(N\cdot L)}{(V\cdot H)}\right\} \\
F &= R_0 + (1-R_0)(1-(V\cdot H))^5,
\end{aligned}
\tag{4.1}
$$

onde $m$ é a inclinação da raiz quadrada média da função de distribuição de microfacetas de Backmann (BECKMANN; SPIZZICHINO, 1987).

A segunda, por sua vez, é obtida pelo modelo de refletância apresentado por Ashikhmin e Shirley (ASHIKHMIN; SHIRLEY, 2000) simplificado por Ngan et al. (NGAN; DURAND; MATUSIK, 2005b), representado pela Equação 4.2, onde foi implementado apenas o termo responsável pela reflexão especular. Essa função foi implementada com o objetivo de comparar com a sua expressão original, já existente na ferramenta, buscando avaliar a viabilidade de usar a expressão com uma complexidade menor. Ela é bastante útil para representar a aparência de metais, materiais de superfícies polidas, superfícies puramente difusas, entre outros materiais (ASHIKHMIN; SHIRLEY, 2000).

$$f_r(L,V) = \frac{n+1}{8\pi}\frac{(N\cdot H)^n}{(V\cdot H)max((N\cdot L),(N\cdot V))}F, \tag{4.2}$$

onde $n$ controla a forma do lóbulo especular.

As outras duas funções de refletância que foram inseridas no PBRT são oriundas do trabalho desenvolvido por Brady et al. (BRADY et al., 2014), que descreveram novas funções de refletância obtidas pelo *framework* genBRDF. Esse *framework* aprende novos modelos de BRDF analíticos através da Programação Genética (do inglês, Genetic Programming - GP). Segundo os autores, esses novos modelos de BRDF são compactos e mais precisos do que as atuais alternativas do estado da arte.

Dentre as funções apresentadas por Brady et al. foram escolhidas duas que são resultados do experimento da função de distribuição normal (do inglês, Normal Distribution Function Experiment - NDF Experiment). Nesse experimento, utilizaram a expressão do modelo da BRDF de Walter et al. (WALTER et al., 2007), e alteraram os termos da distribuição normal e da geometria do modelo original, ou seja, o $n_r(L,V)$ da Equação 4.3. Segundo os autores, a NDF expressa nas novas funções de refletância, proporciona uma grande variedade de fomas para o lóbulo especular e uma aproximação mais precisa dos dados medidos em comparação com modelos atuais.

$$f_r(L,V) = \frac{K_d}{\pi} + K_s\frac{n_r(L,V)F}{4(V\cdot N)(L\cdot N)}. \tag{4.3}$$

Com a alteração do termo $n_r(L,V)$, Brady et al. definiram duas expressões, uma utilizada para aproximar materiais dielétricos e a outra para os metais. A primeira, utilizada para representar dielétricos, é definida pela Equação 4.4 e ao simplificá-la tem-se a BRDF apresentada na Equação 4.5.

$$\begin{aligned} n_r(L,V) &= \frac{(V\cdot N)(L\cdot N)}{(L\cdot H)}D' \\ D' &= e^{-\left(\frac{\tan\delta}{\beta^2}\right)^{\alpha}} \end{aligned} \tag{4.4}$$

$$f_r(L,V) = \frac{K_d}{\pi} + K_s\frac{D'F}{4(L\cdot H)}. \tag{4.5}$$

A segunda expressão, utilizada para aproximar os metais, é definida pela Equação 4.6.

$$\begin{aligned} n_r(L,V) &= \frac{D'G'}{\alpha^{-2}\beta^{-2}\tan^2\delta + c} \\ G' &= min\left\{1+\beta^2(1+\cos\delta), 2\frac{(L\cdot N)(V\cdot N)}{(L\cdot H)}, \frac{1}{\beta^2\tan\delta}\right\}, \end{aligned} \tag{4.6}$$

onde $c = \cos^4(1) = 0.9994$, mas na implementação foi atribuído $c = 1$.

## 4.3 Materiais Implementados

Com o objetivo de visualizar os resultados das funções de refletância inseridas, foram implementados materiais que são atributos do arquivo de entrada do PBRT, que chamam as funções utilizadas para definir como eles vão dispersar a luz ao interagir com a mesma. Neste contexto, foram inseridos os seguinte materiais na ferramenta de renderização PBRT: *dielectrics* (Seção 4.3.1), *metalBrady* (Seção 4.3.2), e *substrateASI* (Seção 4.3.3), onde o primeiro destes permite escolher uma entre duas funções de refletância implementadas. Ambos materiais *dielectrics* e *metalBrady*, possuem funções de refletância oriundas de Brady et al. (BRADY et al., 2014) que utilizam materiais medidos a partir da base de dados do MERL (MATUSIK et al., 2003) para auxiliar na estimação dessas funções. Ressalta-se que os materiais implementados são baseados nos já existentes, porém são adequados para as funções de refletância implementadas.

### 4.3.1 Dielectrics

Para a implementação do *dielectrics*, material baseado no *plastic*, foi utilizada a função de refletância apresentada pelo modelo de Cook e Torrance 4.7 e a de Brady et al. para dielétricos 4.8, onde a função utilizada pode ser escolhida de acordo com os parâmetros passados pelo usuário no arquivo pbrt. Já o *plastic* utiliza o modelo de Torrance e Sparrow (TORRANCE; SPARROW, 1967) (Equação 4.9).

$$f_r(L,V) = \frac{K_d}{\pi} + K_s \frac{F}{\pi} \frac{DG}{(N \cdot L)(N \cdot V)} \tag{4.7}$$

$$f_r(L,V) = \frac{K_d}{\pi} + K_s \frac{D'F}{4(L \cdot H)} \tag{4.8}$$

$$f_r(p,V,L) = \frac{K_d}{\pi} + K_s \frac{DGF}{4\cos\theta_o \cos\theta_i}, \tag{4.9}$$

onde $D$ da função de Torrance e Sparrow é dado pela Equação:

$$D = \frac{e^{-\tan^2\theta_H(\cos^2\phi_H/\alpha_x^2 + \sin^2\phi_H/\alpha_y^2)}}{\pi\alpha_x\alpha_y\cos^4\theta_H}, \tag{4.10}$$

onde $\alpha_x$ é dado para microfacetas orientadas perpendicularmente ao eixo $x$ e $\alpha_y$ para o eixo $y$. Para mais informações, veja o artigo de Torrance e Sparrow (TORRANCE; SPARROW, 1967).

Comparado ao *plastic*, o material *dielectrics* possui três novos parâmetros ($\alpha$, $\beta$ e $m$). Estes foram adicionados por causa da utilização dos mesmos nas expressões. Por padrão, o *dielectrics* utiliza a função de Cook e Torrance com $m = 1.0$, mas o usuário pode escolher outro valor para $m$ através do arquivo pbrt. Ao determinar o valor de $m$ é importante saber que, quanto mais próximo de zero ele for maior será a amplitude do lóbulo especular. Para escolher a função de Brady et al. o usuário deverá informar os valores dos parâmetros $\alpha$ e $\beta$ no arquivo pbrt,

ambos devem ser maiores que zero. O *dielectrics*, ao contrário do *plastic*, não inclui parâmetros referentes a rugosidade, e recebeu esse nome devido aos modelos de refletância usados serem apropriados para materiais dielétricos que não se restringem somente aos plásticos.

Abaixo segue a implementação do material *dielectrics* (Código 1), e em seguida o trecho correspondente à implementação das funções de refletâncias utilizadas por ele (Códigos 2 e 3 ).

```

#include "materials/dielectrics.h"
#include "spectrum.h"
#include "reflection.h"
#include "paramset.h"
#include "texture.h"
#include "interaction.h"

namespace pbrt {

void DielectricsMaterial::ComputeScatteringFunctions(
    SurfaceInteraction *si, MemoryArena &arena, TransportMode mode,
    bool allowMultipleLobes) const {
    // Perform bump mapping with _bumpMap_, if present
    if (bumpMap) Bump(bumpMap, si);
    si->bsdf = ARENA_ALLOC(arena, BSDF)(*si);
    // Initialize diffuse component of plastic material
    Spectrum kd = Kd->Evaluate(*si).Clamp();
    if (!kd.IsBlack())
        si->bsdf->Add(ARENA_ALLOC(arena,LambertianReflection)(kd));
    // Initialize specular component of plastic material
    Spectrum ks = Ks->Evaluate(*si).Clamp();
    if (!ks.IsBlack()) {
        Float m_par = m->Evaluate(*si);
        Float alfa_par = alfa->Evaluate(*si);
        Float beta_par = beta->Evaluate(*si);
        if(alfa_par == 0.0 || beta_par == 0.0){
           si->bsdf->Add(ARENA_ALLOC (arena,
              CookTorranceIsotropicReflection)(ks, m_par));
        }
        else{
           si->bsdf->Add(ARENA_ALLOC(arena,
              BradyAIsotropicReflection)(ks, alfa_par,beta_par));
       }
    }
}
```

```
35
36 DielectricsMaterial *CreateDielectricsMaterial(const TextureParams
      &mp) {
37     std::shared_ptr<Texture<Spectrum>> Kd =
38         mp.GetSpectrumTexture("Kd", Spectrum(0.25f));
39     std::shared_ptr<Texture<Spectrum>> Ks =
40         mp.GetSpectrumTexture("Ks", Spectrum(0.25f));
41     std::shared_ptr<Texture<Float>> bumpMap =
42         mp.GetFloatTextureOrNull("bumpmap");
43     std::shared_ptr<Texture<Float>> m =
44         mp.GetFloatTexture("m", 1.0f);
45     std::shared_ptr<Texture<Float>> alfa =
46         mp.GetFloatTexture("alfa", 0.0f);
47     std::shared_ptr<Texture<Float>> beta =
48         mp.GetFloatTexture("beta", 0.0f);
49     return new DielectricsMaterial(Kd, Ks, bumpMap, m, alfa, beta);
50 }
51 }  // namespace pbrt
```

Código 1 – Implementação em C++ do material *dielectrics*.

```
1
2 CookTorranceIsotropicReflection::
3 CookTorranceIsotropicReflection(const Spectrum &Rs, const Float m)
4     : BxDF(BxDFType(BSDF_REFLECTION | BSDF_GLOSSY)),
5       Rs(Rs),
6       m(m){}
7 Spectrum CookTorranceIsotropicReflection::
8 f(const Vector3f &wo, const Vector3f &wi) const {
9     Float cosThetaO = AbsCosTheta(wo), cosThetaI = AbsCosTheta(wi);
10    Vector3f wh = wi + wo;
11    if (cosThetaI == 0 || cosThetaO == 0) return Spectrum(0.);
12    if (wh.x == 0 && wh.y == 0 && wh.z == 0) return Spectrum(0.);
13    wh = Normalize(wh);
14    Spectrum F = SchlickFresnel(Dot(wi, wh));
15    Float NdotWh = AbsCosTheta(wh);
16    Float NdotWo = AbsCosTheta(wo);
17    Float NdotWi = AbsCosTheta(wi);
18    Float WOdotWh = AbsDot(wo, wh);
19    Float G = min(1.f, min((2.f * NdotWh * NdotWo / WOdotWh),
20                           (2.f * NdotWh * NdotWi / WOdotWh)));
21    Float tan2Theta = Tan2Theta(wh);
22    if (std::isinf(tan2Theta)) return Spectrum(0.);
```

```
23      Float cos4Theta = Cos2Theta(wh) * Cos2Theta(wh);
24      Float D = std::exp(-(tan2Theta*tan2Theta/m*m)) /(m*m *
            cos4Theta);
25      return (Rs * D * G * F) /
26              (Pi* NdotWi * NdotWo);
27  }
28  std::string CookTorranceIsotropicReflection::ToString() const {
29      return std::string("[ CookTorranceIsotropicReflection Rs: ") +
            Rs.ToString() + std::string(" ]");
30  }
```

Código 2 – Implementação em C++ da função de refletância de Cook e Torrance utilizada pelo material *dielectrics*.

```
1   BradyAIsotropicReflection::BradyAIsotropicReflection
2   (const Spectrum &Rs, const Float alfa, const Float beta)
3       : BxDF(BxDFType(BSDF_REFLECTION | BSDF_GLOSSY)),
4         Rs(Rs),
5         alfa(alfa),
6         beta(beta){}
7   Spectrum BradyAIsotropicReflection::
8   f(const Vector3f &wo, const Vector3f &wi) const {
9       Float cosThetaO = AbsCosTheta(wo), cosThetaI = AbsCosTheta(wi);
10      Vector3f wh = wi + wo;
11      if (cosThetaI == 0 || cosThetaO == 0) return Spectrum(0.);
12      if (wh.x == 0 && wh.y == 0 && wh.z == 0) return Spectrum(0.);
13      wh = Normalize(wh);
14      Spectrum F = SchlickFresnel(Dot(wi, wh));
15      Float WidotWh = AbsDot(wi, wh);
16      Float tanTheta = TanTheta(wh);
17      if (std::isinf(tanTheta) || tanTheta < 0.0) return Spectrum(0.)
            ;
18      Float expr = tanTheta/(beta*beta);
19      Float powalfa = std::pow(expr,alfa);
20      Float D = std::exp(-powalfa);
21      Spectrum rspecular = (D * F) / (4.0*WidotWh);
22      return (Rs * D *F) /
23              (4.0*WidotWh);
24  }
25  std::string BradyAIsotropicReflection::ToString() const {
26      return std::string("[ BradyAIsotropicReflection Rs: ")
27                      + Rs.ToString() + std::string(" ]");
```

```
28 }
```

Código 3 – Implementação em C++ da função de refletância de Brady et al. para materiais dielétricos utilizada pelo material *dielectrics*.

### 4.3.2 MetalBrady

O *metalBrady* é baseado no material *metal* original, porém se diferem bastante, uma vez que vários parâmetros não foram implementados por não serem necessários, tais como rugosidade, índice de refração, coeficiente de absorção, entre outros. Além disso, a função de refletância utilizada pelo metalBrady é baseada na formulação de Brady et al. para metais (Equação 4.11), e a função do *metal* original é dada pelo modelo de Torrance e Sparrow com alteração da função de Fresnel (Equação 4.12).

$$n_r(L,V) = \frac{D'G'}{\alpha^{-2}\beta^{-2}\tan^2\delta + c}$$
$$G' = min\left\{1+\beta^2(1+\cos\delta), 2\frac{(L\cdot N)(V\cdot N)}{(L\cdot H)}, \frac{1}{\beta^2\tan\delta}\right\} \tag{4.11}$$

$$f_r(p,V,L) = \frac{K_d}{\pi} + K_s\frac{DGFresnelCondutor}{4\cos\theta_o\cos\theta_i}, \tag{4.12}$$

onde FresnelCondutor representa o $F$ da Equação 4.9, e é uma implementação específica da ferramenta PBRT.

Na implementação do *metalBrady* foram inseridos os parâmetros $\alpha$ e $\beta$, que por padrão possuem os valores $\alpha = 1.0$ e $\beta = 0.5$. Da mesma forma que o *dielectrics*, esses valores podem ser alterados pelo usuário quando definidos no arquivo pbrt. O Código 4 ilustra a implementação do *metalBrady*; o trecho correspondente à implementação da função de refletância utilizada é apresentado no Código 5.

```
1  #include "materials/metalBrady.h"
2  #include "reflection.h"
3  #include "paramset.h"
4  #include "texture.h"
5  #include "interaction.h"
6
7  namespace pbrt {
8
9  // MetalBradyMaterial Method Definitions
10 MetalBradyMaterial::MetalBradyMaterial(const
       std::shared_ptr<Texture<Spectrum>> &Kd,
```

```
11          const std::shared_ptr<Texture<Spectrum>> &Ks,
12          const std::shared_ptr<Texture<Float>> &bumpMap,
13          const std::shared_ptr<Texture<Float>> &alfa,
14          const std::shared_ptr<Texture<Float>> &beta)
15      : Kd(Kd),
16        Ks(Ks),
17        bumpMap(bumpMap),
18        alfa(alfa),
19        beta(beta){}
20  void MetalBradyMaterial::ComputeScatteringFunctions(
        SurfaceInteraction *si, MemoryArena &arena, TransportMode mode,
        bool allowMultipleLobes) const {
21      // Perform bump mapping with _bumpMap_, if present
22      if (bumpMap) Bump(bumpMap, si);
23      si->bsdf = ARENA_ALLOC(arena, BSDF)(*si);
24      // Initialize diffuse component of metlaBrady material
25      Spectrum kd = Kd->Evaluate(*si).Clamp();
26      if (!kd.IsBlack())
27          si->bsdf->Add(ARENA_ALLOC(arena,LambertianReflection)(kd));
28      // Initialize specular component of plastic material
29      Spectrum ks = Ks->Evaluate(*si).Clamp();
30      if (!ks.IsBlack()) {
31      Float alfa_par = alfa->Evaluate(*si);
32      Float beta_par = beta->Evaluate(*si);
33      si->bsdf->Add(ARENA_ALLOC
34      (arena, BradyMetalIsotropicReflection)
35      (ks, alfa_par,beta_par));
36      }
37  }
38
39  MetalBradyMaterial *CreateMetalBradyMaterial(const TextureParams &
        mp) {
40      std::shared_ptr<Texture<Spectrum>> Kd =
41          mp.GetSpectrumTexture("Kd", Spectrum(0.25f));
42      std::shared_ptr<Texture<Spectrum>> Ks =
43          mp.GetSpectrumTexture("Ks", Spectrum(0.25f));
44      std::shared_ptr<Texture<Float>> bumpMap =
45          mp.GetFloatTextureOrNull("bumpmap");
46      std::shared_ptr<Texture<Float>> alfa =
47          mp.GetFloatTexture("alfa", 1.0f);
48      std::shared_ptr<Texture<Float>> beta =
49          mp.GetFloatTexture("beta", .5f);
```

```
50      return new MetalBradyMaterial(Kd, Ks, bumpMap, alfa, beta);
51  }
52  }  // namespace pbrt
```

Código 4 – Implementação em C++ do material *metalBrady*.

```
1   BradyMetalIsotropicReflection::
2   BradyMetalIsotropicReflection
3   (const Spectrum &Rs, const Float alfa, const Float beta)
4       : BxDF(BxDFType(BSDF_REFLECTION | BSDF_GLOSSY)),
5         Rs(Rs),
6         alfa(alfa),
7         beta(beta){}
8   Spectrum BradyMetalIsotropicReflection::
9   f(const Vector3f &wo, const Vector3f &wi) const {
10      Float cosThetaO = AbsCosTheta(wo), cosThetaI = AbsCosTheta(wi);
11      Vector3f wh = wi + wo;
12      if (cosThetaI == 0 || cosThetaO == 0) return Spectrum(0.);
13      if (wh.x == 0 && wh.y == 0 && wh.z == 0) return Spectrum(0.);
14      wh = Normalize(wh);
15      Spectrum F = SchlickFresnel(Dot(wi, wh));
16      Float NdotWh = AbsCosTheta(wh);
17      Float NdotWo = AbsCosTheta(wo);
18      Float NdotWi = AbsCosTheta(wi);
19      Float WOdotWh = AbsDot(wo, wh);
20      Float WidotWh = AbsDot(wi, wh);
21      Float cosThetah = AbsCosTheta(wh);
22      Float tanTheta = TanTheta(wh);
23      if (std::isinf(tanTheta) || tanTheta < 0.0) return Spectrum(0.)
            ;
24      Float expr = tanTheta/(beta*beta);
25      Float powalfa = std::pow(expr,alfa);
26      Float D = std::exp(-powalfa);
27      Float G = min(1.f + (beta*beta*(1.f + cosThetah)),
28                            min(((2.f * NdotWi * NdotWo) / WidotWh),
29                              1.f / (beta*beta*tanTheta)));
30      Float nr = (D*G)/(((1.f/alfa)*(1.f/alfa)*(1.f/beta)*(1.f/beta)*
            tanTheta*tanTheta)+1.f);
31      return (Rs*nr*F)/(4*NdotWi*NdotWo);
32  }
33
34  std::string BradyMetalIsotropicReflection::ToString() const {
35      return std::string("[ BradyMetalIsotropicReflection Rs: ")
```

```
36      + Rs.ToString() + std::string(" ]");
37  }
```

Código 5 – Implementação em C++ da função de refletância de Brady et al. para metais.

### 4.3.3 SubstrateASI

O último material implementado, *substrateASI*, utiliza a função de refletância obtida através do modelo de refletância de Ashikhmin e Shirley (ASHIKHMIN; SHIRLEY, 2000) simplificado por Ngan et al. (NGAN; DURAND; MATUSIK, 2005b) (Equação 4.13), e tem como base o material original *substrate*, que utiliza a expressão original do modelo de Ashikhmin e Shirley (Equação 4.14). O *substrateASI* difere do original por não possuir parâmetros referentes a rugosidade e por adicionar um novo parâmetro $n$ que é responsável por controlar a forma do lóbulo especular. Por padrão $n = 1.0$, e para ser alterado o usuário deve definir o valor desejado no arquivo pbrt.

$$f_r(L,V) = \frac{n+1}{8\pi} \frac{(N \cdot H)^n}{(V \cdot H) max((N \cdot L),(N \cdot V))} F, \tag{4.13}$$

$$f_r(p,L,V) = \frac{28R_d}{23\pi}(1-R_s)\left(1-\left(1-\frac{(n \cdot L)}{2}\right)^5\right)\left(1-\left(1-\frac{(n \cdot V)}{2}\right)^5\right), \tag{4.14}$$

onde $R_s$ é a cor que especifica a refletância especular incidente na normal, e $R_d$ especifica a refletância difusa. Para mais informações veja o artigo de Ashikhmin e Shirley (ASHIKHMIN; SHIRLEY, 2000).

Detalhes da implementação do material *substrateASI* são apresentados pelo Código 6 e a implementação da função de refletância utilizada é apresentada pelo Código 7.

```
1  #include "materials/substrateasi.h"
2  #include "spectrum.h"
3  #include "reflection.h"
4  #include "paramset.h"
5  #include "texture.h"
6  #include "interaction.h"
7
8  namespace pbrt {
9
10 // SubstrateASIMaterial Method Definitions
11 void SubstrateASIMaterial::ComputeScatteringFunctions(
12     SurfaceInteraction *si, MemoryArena &arena, TransportMode mode,
```

```
13      bool allowMultipleLobes) const {
14      // Perform bump mapping with _bumpMap_, if present
15      if (bumpMap) Bump(bumpMap, si);
16      si->bsdf = ARENA_ALLOC(arena, BSDF)(*si);
17      Spectrum kd = Kd->Evaluate(*si).Clamp();
18      Spectrum ks = Ks->Evaluate(*si).Clamp();
19      Float n_par = n->Evaluate(*si);
20      if (!kd.IsBlack() || !ks.IsBlack()) {
21          si->bsdf->Add(ARENA_ALLOC
22          (arena, ASIsotropicReflection)
23          (ks, n_par));
24      }
25  }
26
27  SubstrateASIMaterial *CreateSubstrateASIMaterial
28  (const TextureParams &mp) {
29      std::shared_ptr<Texture<Spectrum>> Kd =
30          mp.GetSpectrumTexture("Kd", Spectrum(.5f));
31      std::shared_ptr<Texture<Spectrum>> Ks =
32          mp.GetSpectrumTexture("Ks", Spectrum(.5f));
33      std::shared_ptr<Texture<Float>> bumpMap =
34          mp.GetFloatTextureOrNull("bumpmap");
35      std::shared_ptr<Texture<Float>> n =
36          mp.GetFloatTexture("n", 1.0f);
37      return new SubstrateASIMaterial(Kd, Ks, bumpMap, n);
38  }
39
40  }  // namespace pbrt
```

Código 6 – Implementação em C++ do material *substrateASI*.

```
1
2   ASIsotropicReflection::
3   ASIsotropicReflection(const Spectrum &Rs, const Float n)
4       : BxDF(BxDFType(BSDF_REFLECTION | BSDF_GLOSSY)),
5         Rs(Rs),
6         n(n){}
7   Spectrum ASIsotropicReflection::
8   f(const Vector3f &wo, const Vector3f &wi) const {
9       Float cosThetaO = AbsCosTheta(wo), cosThetaI = AbsCosTheta(wi);
10      Vector3f wh = wi + wo;
11      if (cosThetaI == 0 || cosThetaO == 0) return Spectrum(0.);
12      if (wh.x == 0 && wh.y == 0 && wh.z == 0) return Spectrum(0.);
```

```
13      wh = Normalize(wh);
14      Spectrum F = SchlickFresnel(Dot(wi, wh));
15      Float NdotWh = AbsCosTheta(wh);
16      Float NdotWo = AbsCosTheta(wo);
17      Float NdotWi = AbsCosTheta(wi);
18      Float WOdotWh = AbsDot(wo, wh);
19      Float T = max(NdotWi,NdotWo);
20      Float K = ((n+1)/(8.0*Pi));
21      Float Q = std::pow(NdotWh,n) / (WOdotWh * T);
22      return Rs*K*Q*F;
23  }
24
25  std::string ASIsotropicReflection::ToString() const {
26      return std::string("[ ASIsotropicReflection Rs: ") + Rs.
            ToString() + std::string(" ]");
27  }
```

Código 7 – Implementação em C++ da função de refletância de Ashikhmin e Shirley simplificada por Ngan et al. utilizada pelo material *substrateASI*.

# 5 Resultados

Esta seção apresenta as renderizações obtidas com o uso dos materiais implementados na ferramenta de renderização escolhida (PBRT-v3). Além disso, uma análise e comparação sobre os resultados obtidos é feita. Para o desenvolvimento dessas renderizações um conjunto de cenas simples (Seção 5.1) e complexas (Seção 5.2) foram criadas, e seus resultados apresentados. Todas as renderizações desenvolvidas nesta seção foram realizadas em um computador Dekstop com processador Intel Core i3-4130 CPU @ 3.40GHz x 4, 4 GB de memória RAM e gráficos Intel Haswell.

## 5.1 Cenas Simples

As Figuras 13 e 14 ilustram o uso dos materiais *plastic* e *dielectrics* em uma cena simples composta por um modelo de um bule iluminado com uma luz de área infinita. A Figura 13 apresenta resultados obtidos ao renderizar a cena com o material *dielectrics* utilizando o modelo de refletância de Cook e Torrance (COOK; TORRANCE, 1982) em comparação com o *plastic*. Já a Figura 14 faz uma comparação semelhante, diferenciando apenas no *dielectrics* que utiliza o modelo de refletância de Brady et al. (BRADY et al., 2014) para dielétricos. Ambas as figuras mostram que o material *dielectrics* gastou um menor tempo de renderização para gerar a mesma cena em comparação com o *plastic*, e obtém resultados semelhantes. Além disso, esse novo material permite ao usuário a flexibilidade de ajustar a especularidade quando alterado o parâmetro $m$ (para o modelo de Cook e Torrance), e os parâmetros $\alpha$ e $\beta$ (para o modelo de Brady et al.)

(a) tempo: 1387.6*s* (b) tempo: 1312.6*s* (c) tempo: 1307.6*s*

Figura 13 – Cena simples de um bule e uma fonte de luz de área infinita. Renderizado com o material (a) *plastic*, (b) *dielectrics* com $m = 0.7$, (c) *dielectrics* com $m = 1.0$. Em ambas as renderizações (b) e (c), foi utilizado o modelo de Cook e Torrance.

(a) tempo: 1387.6*s* (b) tempo: 1332.6*s* (c) tempo: 1327.6*s*

Figura 14 – Renderização de uma cena simples composta por um bule e uma fonte luz de área infinita. Cena renderizada com o material (a) *plastic*, (b) *dielectrics* com $\alpha = 1.0$ e com $\beta = 0.5$, (c) *dielectrics* com $\alpha = 1.0$ e com $\beta = 1.0$. Para as renderizações (b) e (c) foi utilizado o modelo de Brady et al. para materiais dielétricos.

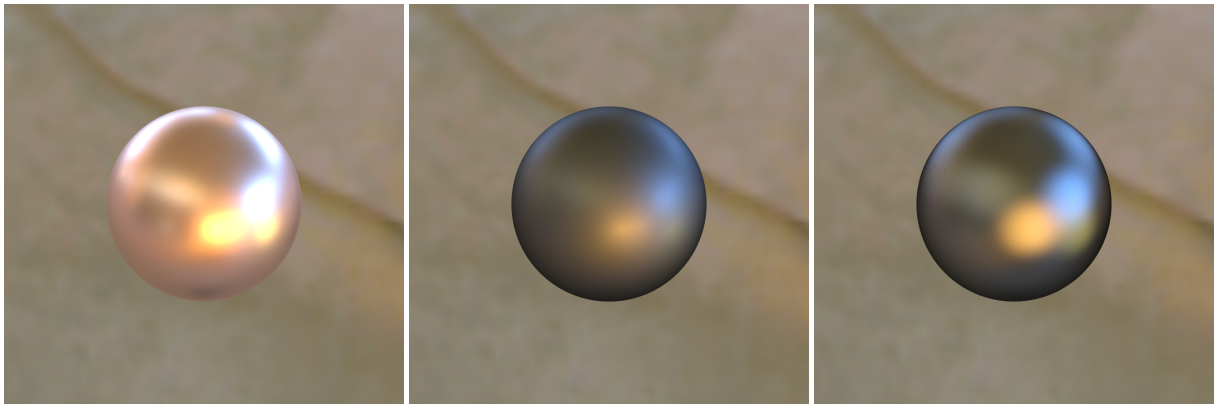
Utilizando materiais mais complexos, duas cenas simples foram desenvolvidas. A primeira composta pelo modelo de um dragão e iluminada com uma luz de área infinita, é apresentada na Figura 15 e ilustra uma comparação entre o material original *substrate* com o inserido *substrateASI* (Seção 4.3.3). Nota-se que, dependendo do valor de $n$, o material *substrateASI* pode custar mais tempo de renderização em comparação com o original, mas obtém picos especulares com mais destaques.

A segunda cena é composta por uma esfera e, assim como a primeira, é iluminada por uma luz de área infinita. A Figura 16 mostra os resultados obtidos na renderização dessa cena, onde os materiais usados foram o *metal* original e o *metalBrady* (Seção 4.3.2). Ressalta-se que o *metal* original possui parâmetros que não foram implementados no material inserido, pois a função de refletância não necessita. No entanto, na Figura 16 nota-se que a aparência obtida com o *metal* original é diferente da obtida com o *metalBrady*, isso ocorre devido aos parâmetros existentes no original interferirem na aparência. Neste caso, não é adequado fazer

uma comparação de tempo de renderização, mas essa informação está disponível para que se saiba o tempo consumido.

(a) tempo: 1352.6$s$ (b) tempo: 1322.6$s$ (c) tempo: 1367.6$s$

Figura 15 – Renderização de uma cena composta por um dragão sob uma fonte de luz de área infinita, renderizada com o material (a) *substrate*, (b) *substrateASI* com $n = 4.0$, (c) *substrateASI* com $n = 8.0$.



(a) tempo: 1598.1$s$ (b) tempo: 1483.2$s$ (c) tempo: 1398.0$s$

Figura 16 – Renderizações de esferas utilizando uma fonte de luz de área infinita e os materiais (a) *metal*, (b) *metalBrady* com $\alpha = 0.5$ e $\beta = 0.5$, (c) *metalBrady* com $\alpha = 3.0$ e $\beta = 0.5$. Nota-se que, ao comparar (b) e (c), a Figura (b) apresenta resultados mais próximos da realidade, pois dar uma impressão de menor contraste.

## 5.2 Cenas Complexas

A renderização realista visa simular uma imagem do mundo real. Neste contexto, uma cena foi desenvolvida por meio da ferramenta PBRT, com o objetivo de simular uma foto de uma

sala de jantar. Esse experimento teve como base uma cena disponível pela própria ferramenta (Figura 17) e imagens do mundo real. Para facilitar o desenvolvimento, a montagem da cena foi feita com a ferramenta Blender (Seção 3.2.2) e, em seguida, a geometria obtida foi exportada para ser utilizada no PBRT.

Figura 17 – Cena disponibilizada pelo PBRT, utilizada como inspiração para o desenvolvimento da cena complexa.

No PBRT, foram definidas as opções de renderização (apresentadas na Seção 2.2.1) e os elementos da descrição da cena (apresentados na Seção 2.2.2) utilizados. A posição da câmera foi estabelecida para três pontos de vistas distintos, um que foca em toda a cena, outro para visualizar a direção da fonte de luz de maior intensidade, e o último ressalta os objetos sobre a mesa. Os outros atributos escolhidos foram: câmera *perspectiva*, amostrador *sobol* com *integer pixelsamples* = 2048, integrador *bdpt* com *integer maxdepth* = 10, e além desses, um conjunto de materiais e texturas. Abaixo, estão todos os nomes dos objetos que compõe a cena com os materiais e seus parâmetros utilizados:

- Paredes - material *uber*, parâmetros: "color Kd"[0.374624 0.374624 0.374624], "color Ks"[0.047366 0.047366 0.047366], "color Kr"[0.000000 0.000000 0.000000];
- Teto - material *uber*, parâmetros: "color Kd"[0.374624 0.374624 0.374624], "color Ks"[0.047366 0.047366 0.047366], "color Kr"[0.000000 0.000000 0.000000];
- Solo - material *uber*, parâmetros: "texture Kdfloor-kd", "rgb Ks"[.04 .04 .04] "float roughness"[.02], "texture bumpmapfloor-disp";
- Quadro - material *matte*, parâmetros: "texture Kdtexturapintura";

- Tapete - material *uber*, parâmetros: "texture Kdcarpet-kd", "rgb Ks"[.04 .04 .04] "float roughness"[.02], "texture bumpmapcarpet-disp";
- Vasculhante - material *plastic*, parâmetros default;
- Lustre - material *matte*, parâmetros: "texture Kdtexturalustre";
- Partes subjacentes do lustre - material *plastic*, parâmetros default;
- Tábua da mesa - material *glass*, parâmetros: "string type"["glass"], "rgb Kr"[0.1 0.1 0.1];
- Pernas da mesa - material *metal*, parâmetros: "string type"["metal"], "spectrum eta spds/K.eta.spd", "spectrum kspds/K.k.spd";
- Almofado das cadeiras - material *uber*, parâmetros: "texture bumpmapscaled-couch-disp", "color Kd"[0.8 .8 .8], "color Ks"[0.1 .1 .1] "float roughness"[.02], "color Kr"[0.000000 0.000000 0.000000];
- Pernas das cadeiras - material *metal*, parâmetros: "string type"["metal"], "spectrum eta spds/K.eta.spd", "spectrum kspds/K.k.spd";
- Xícaras - material *fourier*, parâmetros: "string bsdffilebsdfs/ceramic.bsdf";
- Colheres - material *metal*, parâmetros: "string type"["metal"], "spectrum etaspds/K.eta.spd", "spectrum kspds/K.k.spd";
- Prato - material *metal*, parâmetros: "string type"["metal"], "spectrum etaspds/K.eta.spd", "spectrum kspds/K.k.spd";
- Copos - material *glass*, parâmetros: "string type"["glass"], "rgb Kr"[0.1 0.1 0.1];
- Jarra de suco - material *plastic*, parâmetros: "rgb Kd"[0.1843 0.3098 0.3098]
- Rosa - material *matte*, parâmetros: "texture Kdtexturarosa";
- Caule da rosa - material *matte*, parâmetros: "texture Kdtexturagreen";
- Vaso - material *plastic*, parâmetros: "rgb Kd"[ 0.1 0.1 0.1 ] "rgb Ks"[ 1 1 1 ];
- Pão - material *matte*, parâmetros: "texture Kdtexturapao";
- Tigela - material *substrate*, parâmetros: "rgb Kd"[0.0 0.0 0.0], "rgb Ks"[0.1 0.1 0.1].

Detalhes sobre arquivos que descrevem as geometrias, especificação de texturas, e intensidade da iluminação na cena podem ser vistos no apêndice. Para ilustrar como a cena está iluminada, esferas foram inseridas em posições próximas das fontes de luz na sala vazia, com o objetivo de apresentar a localização das fontes de luz dentro da sala, que no total foram oito

ponto de luz (Figura 18). As iluminações restantes da cena consistem em uma luz de área infinita que utiliza a Figura 19 como mapa de textura para representar uma luz ambiente, e em uma luz distante localizada na direção da janela em que os raios de luz incidem na sala.

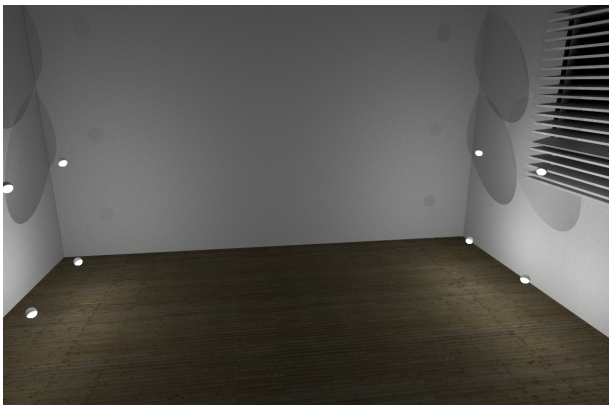


Figura 18 – Iluminação dentro da sala de jantar. Um total de oito luzes pontuais com intensidades diferentes foram utilizadas e espalhadas na sala. As posições de cada uma delas são aproximadamente representadas pelas esferas.



Figura 19 – Imagem de um ambiente para representar iluminação ambiente ao usar como mapa de textura para a luz de área infinita.

Utilizando três posições diferentes da câmera, a sala de jantar foi renderizada buscando analisar vários aspectos da cena. A primeira posição, mostrada na Figura 20, tem como objetivo propor a visualização de todos os objetos da cena. Além disso, a mesma cena foi renderizada com alguns dos seus objetos utilizando materiais que foram inseridos na ferramenta, mostrados na Figura 21. Ao analisar o tempo de renderização das duas figuras, notou-se que a cena que utiliza materiais inseridos consumiu mais tempo. Abaixo são descritos os objetos escolhidos e os materiais inseridos com seus parâmetros utilizados.

- Tigela - material *substrateASI*, parâmetros: "rgb Kd"[0.0 0.0 0.0] "rgb Ks"[0.5 0.5 0.5] "float n"4.0;
- Vaso - material *dielectrics*, parâmetros: "rgb Kd"[ 0.1 0.1 0.1 ], "rgb Ks"[ 0.5 0.5 0.5 ], "float m"[1.0];

- Jarra - material *dielectrics*, parâmetros: "rgb Kd"[0.1843 0.3098 0.3098], "float alfa"[3.0] "float beta"[1.0];
- Prato - material *metalBrady*, parâmetros: "rgb Kd"[0.3 0.3 0.3], "rgb Ks"[1.0 1.0 1.0], "float alfa"0.5 "float beta"0.5;

Figura 20 – Sala de jantar renderizada com os materiais já existentes na ferramenta PBRT. Foi necessário um tempo de renderização de 20538.1$s$ . O parâmetro referente a posição da câmera é dada por *LookAt* 1 −6.9 5.5 0 0 3.5 0 0 1.

Figura 21 – Sala de jantar renderizada utilizando também os materiais inseridos, foi necessário um tempo de renderização de 20638.1$s$. O parâmetro referente a posição da câmera é dada por *LookAt* 1 −6.9 5.5 0 0 3.5 0 0 1.

A segunda posição escolhida para a renderização da cena da sala de jantar busca ilustrar a iluminação que vem da janela e incide na sala, apresentada na Figura 22. Ressalta-se que a iluminação que incide fortemente na parte de trás da cadeira do lado esquerdo é vinda de uma luz pontual que está localizada próxima da cadeira (ver Figura 18), e não da luz distante que sai da janela. Assim, como para a primeira posição da câmera, uma renderização foi feita utilizando os materiais inseridos com os mesmo objetos apresentados anteriormente. A Figura 23 mostra o resultado obtido. Nota-se, ao comparar as duas figuras, que diferente do que ocorreu para a primeira posição da câmera, a segunda posição consumiu um tempo de renderização menor para a cena que contém materiais inseridos do que para a cena que contém somente os já existentes.

A última posição escolhida proporciona um foco especial nos objetos que estão sobre a mesa, com o principal objetivo de propor uma comparação mais precisa entre esse objetos com a cena renderizada utilizando ou não os novos materiais inseridos. As Figuras 24 e 25 mostram os resultados, onde somente a segunda utiliza os materiais inseridos nos objetos escolhidos, já apresentados anteriormente. Apesar da semelhança obtida entre as renderizações, a cena que utilizou os materiais inseridos custou mais tempo de renderização para essa posição da câmera.

Figura 22 – Renderização da sala de jantar buscando ilustrar a iluminação ourinda da janela, e que incide na sala. Na renderização, foi utilizado apenas os materiais já existentes na ferramenta PBRT, e o tempo de renderização consumido foi de 19933.1$s$. Parâmetro usado para determinar a posição da câmera: *LookAt* 5.8 $-$6.9 5.5 0 0 3.5 0 0 1.

Figura 23 – Renderização da sala utilizando também os materiais inseridos, ilustrando a iluminação que vem da janela. O tempo de renderização necessário foi de 19807.1$s$. Parâmetro usado para determinar a posição da câmera: *LookAt* 5.8 $-$6.9 5.5 0 0 3.5 0 0 1.

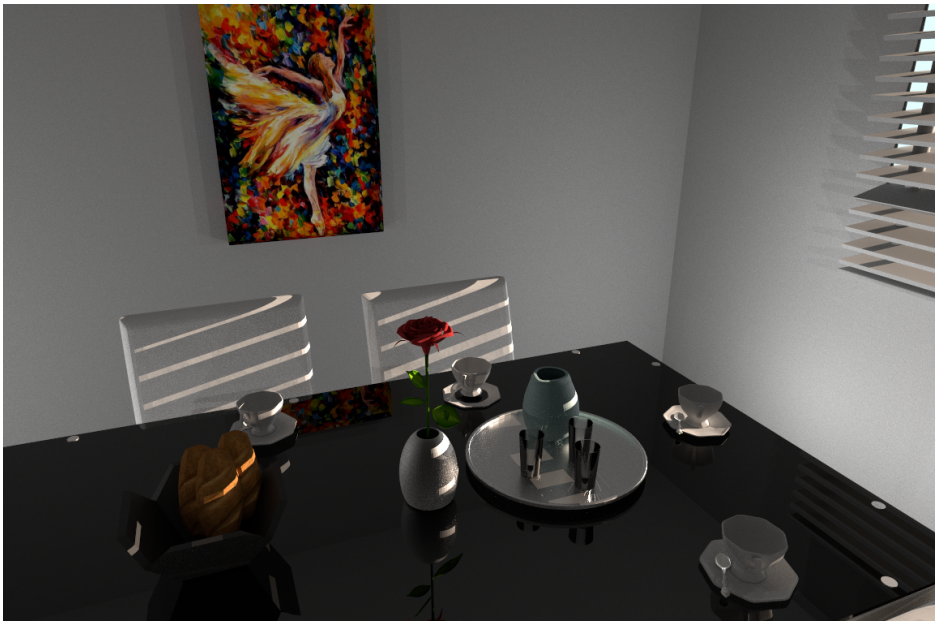


Figura 24 – Sala de jantar renderizada com o foco nos objetos sobre a mesa utilizando somente os materiais já existentes na ferramenta PBRT. Tempo de renderização foi de 22538.2$s$. O parâmetro utilizado pela a câmera foi: *LookAt* 1 −2.5 5.5 0 0 4.5 0 0 1.

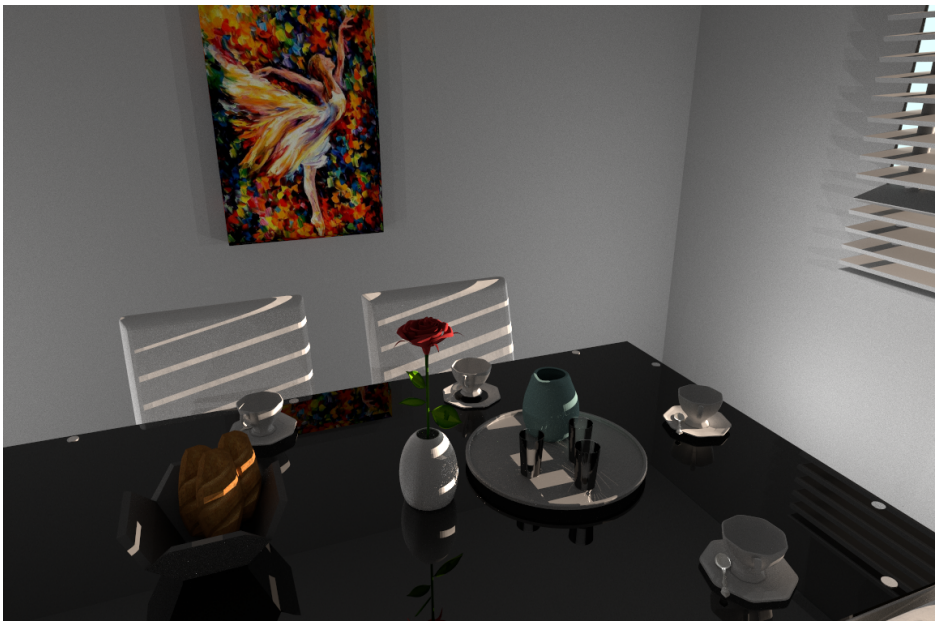


Figura 25 – Sala de jantar renderizada com o foco nos objetos sobre a mesa utilizando também os materiais inseridos. Tempo de renderização foi de 22673.2$s$. O parâmetro utilizado pela a câmera foi: *LookAt* 1 −2.5 5.5 0 0 4.5 0 0 1.

# 6

# Conclusão

Este trabalho aprimorou uma ferramenta de renderização, inserindo funções de refletâncias não contidas na mesma, assim como também, implementou novos materiais que utilizam essas funções para caracterizar a forma como eles refletem a luz. Dessa forma, foi proporcionado um aumento na representatividade de materiais do mundo real, que podem ser utilizados por diversas áreas da computação gráfica, tais como renderização realista, simulação, entre outras. Além disso, foi mostrado o impacto e a importância que uma função de refletância tem na aparência de um material, onde foram feitas comparações de um mesmo material utilizando diferentes funções de refletância.

Para realizar a inserção de funções de refletância em uma ferramenta de renderização foram estudadas diferentes ferramentas, buscando uma que seja estruturada e que possibilite a produção de renderizações próximas da realidade. Neste contexto, a ferramenta PBRT foi a que mais correspondeu a essas características, principalmente por sua implementação em C++ ser bastante estruturada e de propor renderizações realistas. Além disso, informações sobre o PBRT são de fácil acesso, e a disponibilidade de exemplos de cenas simples e complexas também auxiliam bastante no desenvolvimento de uma nova cena.

Ao estudar a ferramenta PBRT foram analisadas as funções existentes na mesma, e buscadas outras para implementar e aumentar a gama de funções de refletância da ferramenta. Dessa forma, o modelo de refletância de Cook e Torrance (COOK; TORRANCE, 1982), Ashikhmin e Shirley (ASHIKHMIN; SHIRLEY, 2000) simplificado por Ngan et al. (NGAN; DURAND; MATUSIK, 2005b), e Brady et al. (BRADY et al., 2014) foram implementados, e para avaliar os resultados foi necessária a implementação dos materiais *dielectrics*, *metalBrady* e *substrateASI* para utilizar essas funções. Dentre os materiais inseridos o *dielectrics* e o *substrateASI* produziram melhores resultados ao simular materiais do mundo real em comparação ao *metalBrady*, que não produziu resultados tão reais quanto os outros materiais similares a ele.

Os materiais inseridos mostram que podem simular bem materiais já existentes na ferramenta, mas também podem propor detalhes no pico especular com mais destaques tal como apresentado pelo *substrateASI*. Além disso, para cenas simples foi mostrado que os materiais inseridos em geral consumiram menos tempo de renderização e produziram resultados semelhantes comparados aos materiais da ferramenta. A exceção foi o material *metalBrady*, cujos resultados diferem do *metal* original. No entanto, para cenas mais complexas os resultados apresentados mostraram na maioria dos casos que cenas complexas utilizando novos materiais necessitam de mais tempo de renderização.

Analisando a representação da aparência de materiais, o modelo de refletância que obteve mais destaque dentre os estudados, é o de Ashikhmin e Shirley (ASHIKHMIN; SHIRLEY, 2000). Ele proporciona a representação da aparência de materiais isotrópicos e anisotrópicos, onde para materiais isotrópicos há também uma expressão simplificada proposta por Ngan et al. (NGAN; DURAND; MATUSIK, 2005b), que neste trabalho também fizeram uma comparação entre alguns modelos da literatura. Além disso, Ngan et al. mostraram que o modelo de Ashikhmin e Shirley simplificado foi o mais promissor dentre os comparados (NGAN; DURAND; MATUSIK, 2005a).

Para trabalhos futuros, sugere-se o estudo de uma métrica que avalie quantitativamente a aparência desses novos materiais, a fim de descobrir o quanto se aproxima de materiais reais, com o objetivo de validar se esses novos materiais realmente aumentam a gama da representatividade de materiais do mundo real proporcionada pela ferramenta. Outra sugestão é a implementação de um material que carrega dados de uma BRDF medida na terceira versão do PBRT. Este material existia na segunda versão do PBRT mas na terceira versão foi substituído pelo material Fourier que apenas carrega dados de uma BSDF medida.

# Referências

ASHIKHMIN, M.; SHIRLEY, P. An Anisotropic Phong BRDF Model. *J. Graph. Tools*, A. K. Peters, Ltd., Natick, MA, USA, v. 5, n. 2, p. 25–32, fev. 2000. ISSN 1086-7651. Disponível em: <http://dx.doi.org/10.1080/10867651.2000.10487522>. Citado 8 vezes nas páginas 23, 35, 37, 38, 39, 48, 61 e 62.

ASHIKMIN, M.; PREMOZE, S.; SHIRLEY, P. A Microfacet-based BRDF Generator. In: *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 2000. (SIGGRAPH '00), p. 65–74. ISBN 1-58113-208-5. Disponível em: <http://dx.doi.org/10.1145/344779.344814>. Citado na página 21.

ASMAIL, C. Bidirectional Scattering Distribution Function (BSDF): A Systematized Bibliography. *Journal of Research of the National Institute of Standards and Technology*, National Institute of Standards and Technology, v. 96, n. 2, p. 215, 1991. Citado na página 20.

BECKMANN, P.; SPIZZICHINO, A. e. Book; Book/Illustrated. *The Scattering of Electromagnetic Waves from Rough Surfaces*. [S.l.]: Norwood, MA : Artech House, 1987. Reprint. Originally published: Oxford [Oxfordshire] ; New York : Pergamon Press, 1963. (International Series of Monographs on Electromagnetic Waves ; v. 4). ISBN 0890062382. Citado na página 39.

BLINN, J. F. Models of Light Reflection for Computer Synthesized Pictures. *SIGGRAPH Comput. Graph.*, ACM, New York, NY, USA, v. 11, n. 2, p. 192–198, jul. 1977. ISSN 0097-8930. Disponível em: <http://doi.acm.org/10.1145/965141.563893>. Citado na página 35.

BORSHUKOV, G.; LEWIS, J. P. Realistic Human Face Rendering for "The Matrix Reloaded". In: *ACM SIGGRAPH 2003 Sketches &Amp; Applications*. New York, NY, USA: ACM, 2003. (SIGGRAPH '03), p. 1–1. Disponível em: <http://doi.acm.org/10.1145/965400.965470>. Citado 2 vezes nas páginas 15 e 34.

BRADY, A. et al. genBRDF: Discovering New Analytic BRDFs with Genetic Programming. *ACM Trans. Graph.*, ACM, New York, NY, USA, v. 33, n. 4, p. 114:1–114:11, jul. 2014. ISSN 0730-0301. Disponível em: <http://doi.acm.org/10.1145/2601097.2601193>. Citado 7 vezes nas páginas 11, 38, 39, 40, 41, 51 e 61.

CARVALHO, B. T. A. d. *Image-Based Appearance Preservation*. Tese (Doutorado) — Universidade Federal do Paraná, 2013. Citado na página 20.

CHRISTOPHE, H.; RYUSUKE, V.; HECHT, F. Towards Bidirectional Path Tracing at Pixar. 2016. Citado na página 34.

COOK, R. L.; TORRANCE, K. E. A Reflectance Model for Computer Graphics. *ACM Trans. Graph.*, ACM, v. 1, n. 1, p. 7–24, jan. 1982. Citado 4 vezes nas páginas 38, 39, 51 e 61.

DOBASHI, Y.; YAMAMOTO, T.; NISHITA, T. Efficient Rendering of Lightning Taking into Account Scattering Effects due to Clouds and Atmospheric Particles. In: *Proceedings Ninth Pacific Conference on Computer Graphics and Applications. Pacific Graphics 2001*. [S.l.: s.n.], 2001. p. 390–399. Citado 2 vezes nas páginas 15 e 33.

FORES, A.; FERWERDA, J.; GU, J. Toward a Perceptually Based Metric for BRDF Modeling. In: Society for Imaging Science and Technology. *Color and Imaging Conference*. [S.l.], 2012. v. 2012, n. 1, p. 142–148. Citado na página 23.

FOUNDATION, B. *Blender*. 2017. Disponível em: <https://www.blender.org/>. Citado 2 vezes nas páginas 8 e 19.

HAVRAN, V.; FILIP, J.; MYSZKOWSKI, K. Perceptually Motivated BRDF Comparison Using Single Image. In: *Proceedings of the Eurographics Symposium on Rendering*. [S.l.]: Eurographics Association, 2016. (EGSR '16), p. 1–12. Citado na página 23.

IBEN, H. et al. Artistic Simulation of Curly Hair. In: *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. New York, NY, USA: ACM, 2013. (SCA '13), p. 63–71. ISBN 978-1-4503-2132-7. Disponível em: <http://doi.acm.org/10.1145/2485895.2485913>. Citado na página 34.

LAFORTUNE, E. P. F. et al. Non-linear Approximation of Reflectance Functions. In: *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1997. (SIGGRAPH '97), p. 117–126. ISBN 0-89791-896-7. Disponível em: <http://dx.doi.org/10.1145/258734.258801>. Citado na página 21.

LAUR, D. M. et al. Ray Tracing for the Movie 'Cars'. *IEEE Symposium on Interactive Ray Tracing 2006*, IEEE Computer Society, Los Alamitos, CA, USA, v. 00, p. 1–6, 2006. Citado 2 vezes nas páginas 15 e 34.

MATUSIK, W. et al. A Data-driven Reflectance Model. *ACM Trans. Graph.*, ACM, New York, NY, USA, v. 22, n. 3, p. 759–769, jul. 2003. ISSN 0730-0301. Citado na página 41.

NGAN, A.; DURAND, F.; MATUSIK, W. Experimental analysis of brdf models. In: *Proceedings of the Eurographics Symposium on Rendering*. [S.l.]: Eurographics Association, 2005. p. 117–226. Citado na página 62.

NGAN, A.; DURAND, F.; MATUSIK, W. *Experimental Analysis of BRDF Models - Supplemental*. [S.l.]: Eurographics Association, 2005. Citado 5 vezes nas páginas 38, 39, 48, 61 e 62.

OREN, M.; NAYAR, S. K. Generalization of Lambert's Reflectance Model. In: *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA: ACM, 1994. (SIGGRAPH '94), p. 239–246. ISBN 0-89791-667-0. Disponível em: <http://doi.acm.org/10.1145/192161.192213>. Citado 2 vezes nas páginas 21 e 35.

PHARR, M.; HUMPHREYS, G. *Physically Based Rendering, Second Edition: From Theory To Implementation*. 2nd. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2010. ISBN 0123750792, 9780123750792. Citado 12 vezes nas páginas 8, 15, 18, 22, 23, 25, 27, 29, 30, 31, 35 e 38.

PHARR, M.; JAKOB, W.; HUMPHREYS, G. *PBRT Rendering System, Version 3*. 2016. Disponível em: <http://www.pbrt.org/fileformat-v3.html>. Citado 5 vezes nas páginas 23, 26, 27, 35 e 38.

PHONG, B. T. Illumination for Computer Generated Pictures. *Commun. ACM*, ACM, New York, NY, USA, v. 18, n. 6, p. 311–317, jun. 1975. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/360825.360839>. Citado 2 vezes nas páginas 21 e 22.

SCHLICK, C. An Inexpensive BRDF Model for Physically-based Rendering. *Computer Graphics Forum*, v. 13, p. 233–246, 1994. Citado na página 22.

SHIRLEY, P.; MARSCHNER, S. *Fundamentals of Computer Graphics*. 3rd. ed. Natick, MA, USA: A. K. Peters, Ltd., 2009. ISBN 1568814690, 9781568814698. Citado 6 vezes nas páginas 8, 15, 21, 22, 23 e 24.

TORRANCE, K. E.; SPARROW, E. M. Theory for Off-Specular Reflection From Roughened Surfaces. *J. Opt. Soc. Am.*, OSA, v. 57, n. 9, p. 1105–1114, Sep 1967. Disponível em: <http://www.osapublishing.org/abstract.cfm?URI=josa-57-9-1105>. Citado 3 vezes nas páginas 21, 35 e 41.

WALTER, B. et al. Microfacet Models for Refraction Through Rough Surfaces. In: *Proceedings of the 18th Eurographics Conference on Rendering Techniques*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2007. (EGSR'07), p. 195–206. ISBN 978-3-905673-52-4. Disponível em: <http://dx.doi.org/10.2312/EGWR/EGSR07/195-206>. Citado na página 40.

WARD, G. J. Measuring and Modeling Anisotropic Reflection. *SIGGRAPH Comput. Graph.*, ACM, New York, NY, USA, v. 26, n. 2, p. 265–272, jul. 1992. ISSN 0097-8930. Disponível em: <http://doi.acm.org/10.1145/142920.134078>. Citado na página 21.

WEBER, J.; PENN, J. Creation and Rendering of Realistic Trees. In: *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA: ACM, 1995. (SIGGRAPH '95), p. 119–128. ISBN 0-89791-701-4. Disponível em: <http://doi.acm.org/10.1145/218380.218427>. Citado na página 33.

WEYRICH, T. et al. Principles of Appearance Acquisition and Representation. *Found. Trends. Comput. Graph. Vis.*, v. 4, n. 2, p. 75–191, fev. 2009. ISSN 1572-2740. Citado 6 vezes nas páginas 8, 16, 19, 20, 21 e 35.

# Apêndices

# APÊNDICE A – Detalhes da Descrição da Cena da Sala de Jantar

## A.1 Opções de Renderização da Cena

Para a criação de uma cena no PBRT é necessário descrever as opções de renderização. Dessa forma o Código 8 apresenta as opções utilizadas na cena da sala de jantar.

```
#LookAt 1 -2.5 5.5  0 0 4.5  0 0 1 # posicao para mostrar a mesa
LookAt 1 -6.9 5.5  0 0 3.5  0 0 1  # posicao para mostrar a cena
   toda
#LookAt 5.8 -6.9 5.5  0 0 3.5  0 0 1 # posicao para mostrar a
   janela

Camera "perspective" "float fov" [50]

Film "image" "string filename" ["sala_jantar.png"]
     "integer xresolution" [1150] "integer yresolution" [760]

Sampler "sobol" "integer pixelsamples"  2048
Integrator "bdpt" "integer maxdepth" [10]

WorldBegin

AttributeBegin

        CoordSysTransform "camera"
        LightSource "infinite" "string mapname" "textures/ceudia.
           png"

AttributeEnd

Scale -1 1 1

Include "materials.pbrt"

Include "geometria_comentada.pbrt"
Include "geometria_descomentada.pbrt"

```

```
29 WorldEnd
```

Código 8 – Arquivo responsável pela descrição das opções de renderização.

## A.2 Descrevendo a Cena

### A.2.1 Formas

Para descrever todos as formas utilizadas na cena um arquivo específico pode ser utilizado para tais descrições, bastando apenas fazer depois a chamada desse arquivo pbrt no arquivo referente as opções de renderização. O Código 9 apresenta todas as formas da sala de jantar, assim como também a maioria das fontes de iluminação.

```
1  #Luz de dentro
2   LightSource "point" "rgb I" [ 10 10 10 ] "point from" [ -5 3.5 1 ]
3          LightSource "point" "rgb I" [ 10 10 10 ] "point from" [ 5
              3.5 1 ]
4          LightSource "point" "rgb I" [ 10 10 10 ] "point from" [ -5
              1 1 ]
5          LightSource "point" "rgb I" [ 5 5 5 ] "point from" [ 5 1 1
              ]
6
7   LightSource "point" "rgb I" [ 5 5 5 ] "point from" [ -5 3.5 3.5 ]
8          LightSource "point" "rgb I" [ 5 5 5 ] "point from" [ 5 3.5
              3.5 ]
9          LightSource "point" "rgb I" [ 5 5 5 ] "point from" [ -5 1
              3.5 ]
10         LightSource "point" "rgb I" [ 5 5 5 ] "point from" [ 5 1
              3.5 ]
11
12 #Luz da janela
13  LightSource "distant" "point from" [32 -1 7 ] "blackbody L" [5000
       10]
14
15 #Paredes
16 AttributeBegin
17         NamedMaterial "corParedes"
18  Shape "plymesh" "string filename" "geometry/parede_lado.ply"
19         Shape "plymesh" "string filename" "geometry/
              parede_lado_janela.ply"
20         Shape "plymesh" "string filename" "geometry/parede_quadro.
              ply"
```

```
21          Shape "plymesh" "string filename" "geometry/parede_fundo.
               ply"
22          Shape "plymesh" "string filename" "geometry/teto.ply"
23
24  NamedMaterial "Parquet"
25          Shape "plymesh" "string filename" "geometry/Solos.ply"
26  NamedMaterial "plastico"
27  Shape "plymesh" "string filename" "geometry/janela.ply"
28 AttributeEnd
29
30 #Quadro
31 AttributeBegin
32          NamedMaterial "pintura"
33 Shape "plymesh" "string filename" "geometry/quadro.ply"
34 AttributeEnd
35
36 #Mesa
37 AttributeBegin
38          NamedMaterial "vidro"
39  Shape "plymesh" "string filename" "geometry/mesa_1.ply"
40   NamedMaterial "inox"
41  Shape "plymesh" "string filename" "geometry/mesa_2.ply"
42 AttributeEnd
43
44 #Cadeiras
45 AttributeBegin
46          NamedMaterial "inox"
47  Shape "plymesh" "string filename" "geometry/cadeira1_1.ply"
48  Shape "plymesh" "string filename" "geometry/cadeira2_1.ply"
49  Shape "plymesh" "string filename" "geometry/cadeira3_1.ply"
50  Shape "plymesh" "string filename" "geometry/cadeira4_1.ply"
51  Shape "plymesh" "string filename" "geometry/cadeira5_1.ply"
52  NamedMaterial "couro"
53  Shape "plymesh" "string filename" "geometry/cadeira1_2.ply"
54  Shape "plymesh" "string filename" "geometry/cadeira2_2.ply"
55  Shape "plymesh" "string filename" "geometry/cadeira3_2.ply"
56  Shape "plymesh" "string filename" "geometry/cadeira4_2.ply"
57  Shape "plymesh" "string filename" "geometry/cadeira5_2.ply"
58 AttributeEnd
59
60 #Xicaras
61 AttributeBegin
```

```
62          NamedMaterial "xicara"
63  Shape "plymesh" "string filename" "geometry/chicara1_1.ply"
64  Shape "plymesh" "string filename" "geometry/chicara2_1.ply"
65  Shape "plymesh" "string filename" "geometry/chicara3_1.ply"
66  Shape "plymesh" "string filename" "geometry/chicara4_1.ply"
67  Shape "plymesh" "string filename" "geometry/chicara5_1.ply"
68  Shape "plymesh" "string filename" "geometry/chicara6_1.ply"
69          NamedMaterial "xicara"
70  Shape "plymesh" "string filename" "geometry/chicara1_2.ply"
71  Shape "plymesh" "string filename" "geometry/chicara2_2.ply"
72  Shape "plymesh" "string filename" "geometry/chicara3_2.ply"
73  Shape "plymesh" "string filename" "geometry/chicara4_2.ply"
74  Shape "plymesh" "string filename" "geometry/chicara5_2.ply"
75  Shape "plymesh" "string filename" "geometry/chicara6_2.ply"
76          NamedMaterial "colher"
77  Shape "plymesh" "string filename" "geometry/chicara1_3.ply"
78  Shape "plymesh" "string filename" "geometry/chicara2_3.ply"
79  Shape "plymesh" "string filename" "geometry/chicara3_3.ply"
80  Shape "plymesh" "string filename" "geometry/chicara4_3.ply"
81  Shape "plymesh" "string filename" "geometry/chicara5_3.ply"
82  Shape "plymesh" "string filename" "geometry/chicara6_3.ply"
83 AttributeEnd
84
85 #bowl e pao
86 AttributeBegin
87          NamedMaterial "browl"
88  Shape "plymesh" "string filename" "geometry/bowl.ply"
89   NamedMaterial "pao"
90  Shape "plymesh" "string filename" "geometry/pao1.ply"
91  Shape "plymesh" "string filename" "geometry/pao2.ply"
92  Shape "plymesh" "string filename" "geometry/pao3.ply"
93 AttributeEnd
94
95 #Vazo e rosa
96 AttributeBegin
97          NamedMaterial "vaso"
98  Shape "plymesh" "string filename" "geometry/vazo.ply"
99          NamedMaterial "rosa"
100  Shape "plymesh" "string filename" "geometry/rosa.ply"
101          NamedMaterial "caule"
102  Shape "plymesh" "string filename" "geometry/caule.ply"
103 AttributeEnd
```

```
104
105 #Prato e copos
106 AttributeBegin
107         NamedMaterial "jarra"
108  Shape "plymesh" "string filename" "geometry/jar.ply"
109         NamedMaterial "inox"
110  Shape "plymesh" "string filename" "geometry/prato.ply"
111         NamedMaterial "vidro"
112  Shape "plymesh" "string filename" "geometry/copo1.ply"
113  Shape "plymesh" "string filename" "geometry/copo2.ply"
114  Shape "plymesh" "string filename" "geometry/copo3.ply"
115 AttributeEnd
116
117 #Lustre
118 AttributeBegin
119         NamedMaterial "lustre"
120  Shape "plymesh" "string filename" "geometry/lustre_1.ply"
121   NamedMaterial "plastico"
122  Shape "plymesh" "string filename" "geometry/lustre_2.ply"
123  NamedMaterial "plastico"
124  Shape "plymesh" "string filename" "geometry/lustre_3.ply"
125 AttributeEnd
126
127 #Tapete
128 AttributeBegin
129         NamedMaterial "woolcarpet"
130  Shape "plymesh" "string filename" "geometry/Tapete_Solo.ply"
131 AttributeEnd
```

Código 9 – Arquivo responsável pelas formas utilizadas na cena.

### A.2.2 Utilizando Somente Materiais já Existentes no PBRT

Quando escolhidas as formas da cena, também deve-se escolher os materiais e as texturas correspondentes para cada uma delas. Dessa forma, o Código 10 apresenta todos os materiais (utilizando somente os já existentes na ferramenta) e texturas utilizadas. Ressalta-se que essas descrições foram feitas em um arquivo pbrt, bastando apenas fazer a chamada dele no arquivo referente as opções de renderização.

```
1 MakeNamedMaterial "plastico"
2 "string type" "plastic"
3
4 #Material das paredes
```

```
5  MakeNamedMaterial "corParedes"
6          "string type" ["uber"]
7          "color Kd" [0.374624 0.374624 0.374624]
8          "color Ks" [0.047366 0.047366 0.047366]
9          "color Kr" [0.000000 0.000000 0.000000]
10
11 #Material do chao
12 Texture "floor-kd" "color" "imagemap" "string filename"
13         "textures/boards+02_d100.png"
14 "float scale" [.15]  "float uscale" [4] "float vscale" [4]
15 Texture "floor-ks" "color" "imagemap" "string filename" "textures/
      boards+02_s2.png"
16 "float scale" [100]  "float uscale" [4] "float vscale" [4] "float
      scale" [.1]
17 Texture "floor-disp" "float" "imagemap" "string filename" "textures
      /boards+02_b030.png"
18 "float scale" [.1]
19 "float uscale" [4] "float vscale" [4]
20
21 MakeNamedMaterial "Parquet"
22    "string type" ["uber"]
23    "texture Kd" "floor-kd"
24    "rgb Ks" [.04 .04 .04] "float roughness" [.02]
25    "texture bumpmap" "floor-disp"
26
27 #Material do tapete
28 Texture "carpet-kd" "color" "imagemap" "string filename" "textures/
      Wool_carpet4.png"
29 "float scale" [.15]  "float uscale" [4] "float vscale" [4]
30 Texture "carpet-ks" "color" "imagemap" "string filename" "textures/
      Wool_carpet_normal.png"
31 "float scale" [100]  "float uscale" [4] "float vscale" [4] "float
      scale" [.1]
32 Texture "carpet-disp" "float" "imagemap" "string filename" "
      textures/Wool_carpet_bmp.png"
33 "float scale" [.1]
34 "float uscale" [4] "float vscale" [4]
35
36 MakeNamedMaterial "woolcarpet"
37    "string type" ["uber"]
38    "texture Kd" "carpet-kd"
39    "rgb Ks" [.04 .04 .04] "float roughness" [.02]
```

```
40    "texture bumpmap" "carpet-disp"
41
42 #Material das cadeiras
43 TransformBegin
44         Texture "couch-disp" "float" "wrinkled"
45         Texture "scaled-couch-disp" "float" "scale" "texture tex1"
              "couch-disp" "float tex2" [.005]
46 TransformEnd
47 MakeNamedMaterial "couro"
48         "texture bumpmap" "scaled-couch-disp"
49         "string type" ["uber"]
50         "color Kd" [0.8 .8 .8]
51         "color Ks" [0.1 .1 .1] "float roughness" [.02]
52         "color Kr" [0.000000 0.000000 0.000000]
53
54 #Material inox
55 MakeNamedMaterial "inox"
56         "string type" ["metal"]
57         "float roughness" [.01]
58     "spectrum eta" "spds/K.eta.spd"
59         "spectrum k" "spds/K.k.spd"
60
61 #Material vidro
62 MakeNamedMaterial "vidro"
63         "string type" ["glass"]
64   "rgb Kr" [0.1 0.1 0.1]
65
66 #Material do vaso
67 MakeNamedMaterial "vaso"
68 "string type" "plastic" "rgb Kd" [ 0.1 0.1 0.1 ] "rgb Ks" [ 1 1 1 ]
69
70 #Material da rosa
71 Texture "texturarosa" "spectrum" "imagemap" "string filename" ["
      textures/rose.png"]
72 MakeNamedMaterial "rosa"
73         "string type" "matte"
74         "texture Kd" "texturarosa"
75
76 #Material do caule da rosa
77 Texture "texturagreen" "spectrum" "imagemap" "string filename" ["
      textures/green.png"]
78 MakeNamedMaterial "caule"
```

```
79          "string type" "matte"
80          "texture Kd" "texturagreen"
81
82  #Material do pao
83  Texture "texturapao" "spectrum" "imagemap" "string filename" ["
        textures/pao.png"]
84  MakeNamedMaterial "pao"
85          "string type" "matte"
86          "texture Kd" "texturapao"
87
88  #Material da xicara
89  MakeNamedMaterial "xicara"
90      "string type" "fourier" "string bsdffile" "bsdfs/ceramic.bsdf"
91
92  #Material da colher
93  MakeNamedMaterial "colher"
94          "spectrum eta" "spds/Ag.eta.spd"
95          "spectrum k" "spds/Ag.k.spd"
96          "float roughness" [0.01]
97          "string type" ["metal"]
98
99  #Material do browl
100 MakeNamedMaterial "browl"
101     "string type" "substrate" "rgb Kd" [0.0 0.0 0.0] "rgb Ks" [0.1
            0.1 0.1]
102
103 #Material para a jarra de suco
104 MakeNamedMaterial "jarra"
105         "string type" "plastic" "rgb Kd" [0.1843 0.3098  0.3098]
106
107 #Material do lustre
108 Texture "texturalustre" "spectrum" "imagemap" "string filename" ["
        textures/large.png"]
109 MakeNamedMaterial "lustre"
110         "string type" "matte"
111         "texture Kd" "texturalustre"
112
113 #Material do quadro
114 Texture "texturapintura" "spectrum" "imagemap" "string filename" ["
        textures/pintura2.png"]
115 MakeNamedMaterial "pintura"
116         "string type" "matte"
```

```
117         "texture Kd" "texturapintura"
```

Código 10 – Arquivo responsável pela descrição dos materiais (apenas os já existentes na ferramenta) e texturas utilizadas na cena.

### A.2.3 Utilizando Materiais já Existentes e Inseridos no PBRT

Um outro arquivo para os materiais foi desenvolvido, este corresponde aos materiais já existentes na ferramenta e aos que foram inseridos. Para utilizar este arquivo basta chamar ele no arquivo referente as opções de renderização. As descrições dos materiais deste arquivo são apresentadas no Código 11.

```
1  MakeNamedMaterial "plastico"
2  "string type" "plastic"
3
4  #Material das paredes
5  MakeNamedMaterial "corParedes"
6          "string type" ["uber"]
7          "color Kd" [0.374624 0.374624 0.374624]
8          "color Ks" [0.047366 0.047366 0.047366]
9          "color Kr" [0.000000 0.000000 0.000000]
10
11 #Material do chao
12 Texture "floor-kd" "color" "imagemap" "string filename" "textures/
      boards+02_d100.png"
13 "float scale" [.15]  "float uscale" [4] "float vscale" [4]
14 Texture "floor-ks" "color" "imagemap" "string filename" "textures/
      boards+02_s2.png"
15 "float scale" [100]  "float uscale" [4] "float vscale" [4] "float
      scale" [.1]
16 Texture "floor-disp" "float" "imagemap" "string filename" "textures
      /boards+02_b030.png"
17 "float scale" [.1]
18 "float uscale" [4] "float vscale" [4]
19
20 MakeNamedMaterial "Parquet"
21    "string type" ["uber"]
22    "texture Kd" "floor-kd"
23    "rgb Ks" [.04 .04 .04] "float roughness" [.02]
24    "texture bumpmap" "floor-disp"
25
26 #Material do tapete
```

```
27 Texture "carpet-kd" "color" "imagemap" "string filename" "textures/
      Wool_carpet4.png"
28 "float scale" [.15]  "float uscale" [4] "float vscale" [4]
29 Texture "carpet-ks" "color" "imagemap" "string filename" "textures/
      Wool_carpet_normal.png"
30 "float scale" [100]  "float uscale" [4] "float vscale" [4] "float
      scale" [.1]
31 Texture "carpet-disp" "float" "imagemap" "string filename" "
      textures/Wool_carpet_bmp.png"
32 "float scale" [.1]
33 "float uscale" [4] "float vscale" [4]
34
35 MakeNamedMaterial "woolcarpet"
36    "string type" ["uber"]
37    "texture Kd" "carpet-kd"
38    "rgb Ks" [.04 .04 .04] "float roughness" [.02]
39    "texture bumpmap" "carpet-disp"
40
41 #Material das cadeiras
42 TransformBegin
43         Texture "couch-disp" "float" "wrinkled"
44         Texture "scaled-couch-disp" "float" "scale" "texture tex1"
              "couch-disp" "float tex2" [.005]
45 TransformEnd
46 MakeNamedMaterial "couro"
47         "texture bumpmap" "scaled-couch-disp"
48         "string type" ["uber"]
49         "color Kd" [0.8 .8 .8]
50         "color Ks" [0.1 .1 .1] "float roughness" [.02]
51         "color Kr" [0.000000 0.000000 0.000000]
52
53 #Material do prato
54 MakeNamedMaterial "prato"
55     "string type" "metalBrady" "rgb Kd" [0.3 0.3 0.3] "rgb Ks" [1.0
          1.0 1.0] "float alfa" 0.5 "float beta" 0.5
56
57 #Material inox
58 MakeNamedMaterial "inox"
59         "string type" ["metal"]
60         "float roughness" [.01]
61     "spectrum eta" "spds/K.eta.spd"
62         "spectrum k" "spds/K.k.spd"
```

```
63
64 #Material vidro
65 MakeNamedMaterial "vidro"
66         "string type" ["glass"]
67   "rgb Kr" [0.1 0.1 0.1]
68
69 #Material do vaso
70 MakeNamedMaterial "vaso"
71 "string type" "dielectrics" "rgb Kd" [ 0.1 0.1 0.1 ] "rgb Ks" [ 0.5
       0.5 0.5 ] "float m" [1.0]
72
73 #Material da rosa
74 Texture "texturarosa" "spectrum" "imagemap" "string filename" ["
      textures/rose.png"]
75 MakeNamedMaterial "rosa"
76         "string type" "matte"
77         "texture Kd" "texturarosa"
78
79 #Material do caule da rosa
80 Texture "texturagreen" "spectrum" "imagemap" "string filename" ["
      textures/green.png"]
81 MakeNamedMaterial "caule"
82         "string type" "matte"
83         "texture Kd" "texturagreen"
84
85 #Material do pao
86 Texture "texturapao" "spectrum" "imagemap" "string filename" ["
      textures/pao.png"]
87 MakeNamedMaterial "pao"
88         "string type" "matte"
89         "texture Kd" "texturapao"
90
91 #Material da xicara
92 MakeNamedMaterial "xicara"
93     "string type" "fourier" "string bsdffile" "bsdfs/ceramic.bsdf"
94
95 #Material da colher
96 MakeNamedMaterial "colher"
97         "spectrum eta" "spds/Ag.eta.spd"
98         "spectrum k" "spds/Ag.k.spd"
99         "float roughness" [0.01]
100        "string type" ["metal"]
```

```
101
102 #Material do browl
103 MakeNamedMaterial "browl"
104    #"string type" "substrateasi" "rgb Kd" [1.0 1.0 1.0] "rgb Ks"
          [1.0 1.0 1.0] "float n" 4.0
105    "string type" "substrateasi" "rgb Kd" [0.0 0.0 0.0] "rgb Ks"
          [0.5 0.5 0.5] "float n" 4.0
106
107 #Material para a jarra de suco
108 MakeNamedMaterial "jarra"
109         "string type" "dielectrics" "rgb Kd" [0.1843 0.3098
              0.3098] "float alfa" [3.0] "float beta" [1.0]
110
111 #Material do lustre
112 Texture "texturalustre" "spectrum" "imagemap" "string filename" ["
       textures/large.png"]
113 MakeNamedMaterial "lustre"
114         "string type" "matte"
115         "texture Kd" "texturalustre"
116
117 #Material do quadro
118 Texture "texturapintura" "spectrum" "imagemap" "string filename" ["
       textures/pintura2.png"]
119 MakeNamedMaterial "pintura"
120         "string type" "matte"
121         "texture Kd" "texturapintura"
```

Código 11 – Arquivo responsável pela descrição dos materiais (existentes e novos) e texturas utilizadas na cena.